JOHNSON GRANT
IN-61
7302
P-87

RESEARCH IN SOFTWARE ALLOCATION FOR ADVANCED MANNED MISSION COMMUNICATIONS AND TRACKING SYSTEMS FINAL REPORT

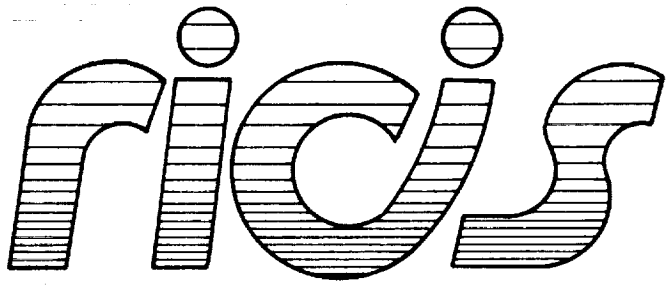
Tom Warnagiris
Bill Wolff
Antone Kusmanoff

Southwest Research Institute

December 1990

Cooperative Agreement NCC 9-16
Research Activity No. AI.1

NASA Johnson Space Center
Engineering Directorate
Tracking and Communications Division



Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information systems in 1986 to encourage NASA Johnson Space Center and local industry to actively support research in the computing and information sciences. As part of this endeavor, UH-Clear Lake proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a three-year cooperative agreement with UH-Clear Lake beginning in May, 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The mission of RICIS is to conduct, coordinate and disseminate research on computing and information systems among researchers, sponsors and users from UH-Clear Lake, NASA/JSC, and other research organizations. Within UH-Clear Lake, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business, Education, Human Sciences and Humanities, and Natural and Applied Sciences.

Other research organizations are involved via the "gateway" concept. UH-Clear Lake establishes relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research.

A major role of RICIS is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. Working jointly with NASA/JSC, RICIS advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research, and integrates technical results into the cooperative goals of UH-Clear Lake and NASA/JSC.

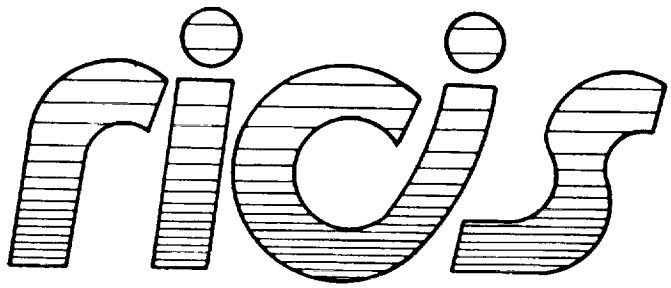
RESEARCH IN SOFTWARE ALLOCATION FOR ADVANCED MANNED MISSION COMMUNICATIONS AND TRACKING SYSTEMS FINAL REPORT

Tom Warnagiris
Bill Wolff
Antone Kusmanoff

Southwest Research Institute
December 1990

Cooperative Agreement NCC 9-16
Research Activity No. AI.1

NASA Johnson Space Center
Engineering Directorate
Tracking and Communications Division



Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Tom Warnagiris, Bill Wolff and Dr. Antone Kusmanoff of the Southwest Research Institute. Dr. Terry Feagin and Dr. T. F. Leibfried served as RICIS research representatives.

Funding has been provided by the Engineering Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Oron Schmidt, of the Communications Performance and Integration Branch, Tracking and Communications Division, Engineering Directorate, NASA/JSC.

The views and conclusions contained in this report are those of the authors and should not be interpreted as representative of the official policies, either express or implied, of NASA or the United States Government.

FINAL REPORT

RESEARCH IN SOFTWARE ALLOCATION FOR ADVANCED MANNED MISSION COMMUNICATIONS AND TRACKING SYSTEMS

FINAL REPORT

RESEARCH IN SOFTWARE ALLOCATION FOR ADVANCED MANNED MISSION COMMUNICATIONS AND TRACKING SYSTEMS

Subcontract No. 079

Prepared for:

NASA Johnson Space Center
under subcontract to the University
of Houston-Clear Lake
Houston, Texas 77058

Prepared by:

Southwest Research Institute

6220 Culebra Road

P. O. Drawer 28510

San Antonio, Texas 78228-0510

FINAL REPORT
FOR THE RESEARCH IN SOFTWARE ALLOCATION FOR ADVANCED MISSION COMMUNICATIONS AND TRACKING SYSTEMS

EXECUTIVE SUMMARY

An assessment of the planned processing hardware and software/firmware for the Communications and Tracking System of the Space Station Freedom (SSF) was performed by Southwest Research Institute. The intent of the assessment was to determine the optimum distribution of software/firmware in the processing hardware for maximum throughput with minimum required memory. As a product of the assessment process an assessment methodology was to be developed that could be used for similar assessments of future manned spacecraft system designs.

The assessment process was hampered by changing requirements for the Space Station. As a result, the initial objective of determining the optimum software/firmware allocation was not fulfilled, but several useful conclusions and recommendations resulted from the assessment. It was concluded that the assessment process would not be completely successful for a system with changing requirements. It was also concluded that memory requirements and hardware requirements were being modified to fit as a consequence of the change process, and although throughput could not be quantitized, potential problem areas could be identified. Finally, inherent flexibility of the system design was essential for the success of a system design with changing requirements.

Recommendations resulting from the assessment included development of common software for some embedded controller functions, reduction of embedded processor requirements by hardwiring some ORUs to make better use of processor capabilities, and improvement in communications between software development personnel to enhance the integration process.

Lastly, a critical observation was made regarding the software development process for the SSF. It was noted that the various hardware/software integration tasks did not appear to be addressed in the design process to the degree necessary for successful satisfaction of the system requirements.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

LIST OF TABLES

1.0 INTRODUCTION

Because space systems have a fixed quantity of hardware resources available, the resources must be adequate for all foreseeable requirements. In many cases, the hardware contains one or more processors controlled by software that actually determines how well the hardware meets the operational requirements for the spacecraft. This software must be adequately accommodated in available spacecraft processor memory.

To accomplish this, a method is needed to efficiently allocate the software to available, yet limited, processor memory. This is a general need for all space processor-controlled systems, but for manned missions is it somewhat different, because the hardware can be modified in space. To develop an assessment method that would efficiently allocate software to the available memory for manned missions, the Space Station Freedom (SSF) Communications and Tracking System (C&TS) was selected as an example system.

1.1 Project Description

The intent of selecting the SSF C&TS was both to develop the assessment methods and to provide useful assessment results to the ongoing SSF C&T design process. Initial goals for the assessment process, specific to the C&TS, were to determine the software allocation that maximized the system throughput while minimizing the hardware required. It was planned that while performing an assessment with these goals, an assessment methodology would evolve that would have application to similar systems on future manned mission spacecraft.

1.2 Initial Project Plan

Since no known assessment for a system of this type had been previously performed, no framework, or prior work could be reviewed to provide a starting point for the assessment process. A plan was needed as a starting point. To begin the process, a draft assessment plan based on an investigative approach was prepared and submitted to JSC personnel early in the program. As the assessment process evolved the plan was updated to form the outline for the assessment methodology (see Appendix A for the final revision).

The initial project plan included descriptions for the following assessment tasks:

1. Meetings - The belief was that by meeting regularly with Johnson Space Center (JSC), University of Houston-Clear Lake (UHCL), and contractor personnel knowledgeable of the C&TS system, a full understanding of the requirements and current status information about the ongoing development process could be obtained.

2. Development of Methodology - With the information obtained during the meetings, the initial project plan could be updated to better reflect the C&TS requirements and the current state of the system design.

3. Data Collection - Collection of data would be the first major step of the assessment process. It was initially anticipated that outside documentation from other NASA agencies and current contractor correspondence would be available in addition to documentation at the JSC.

4. Perform Assessment - With documentation and understanding of the current state of the C&TS design in hand, the parameters in need of assessment could be determined and the assessment process proceed.

5. Oral Presentation - An oral report was planned for presentation toward the end of the assessment process to provide the JSC and UHCL an overview of the assessment and assessment results.

6. Final Report - A final report would be prepared detailing the assessment process. The results would include all findings with conclusions, and recommendations. In addition, it would include a methodology suitable for application to similar software assessments of other manned mission systems.

1.3 Data Collection

The data collection process was used to ascertain the current status of the C&TS design. Unfortunately, current information was difficult to obtain from the cognizant personnel and documentation available.

1.3.1 Interviews with C&T Personnel

Initial interviews were held with key personnel from the JSC Communication and Tracking Department. Through these interviews it was learned that the day to day progress on the C&TS design was in the hands of the prime contractor and the subcontractors. What current information could be gleaned by NASA JSC was through infrequent design reviews and periodic submittal of contractual documents, such as software specifications and hardware descriptions.

In many cases JSC personnel were aware of directed changes that were to be addressed by the contractors, but they had no official information regarding contractor decisions and actions in response to the changes until a document was received or a meeting was held covering the topic. C&T personnel were attending meetings on other SSF systems, such as Data Management System (DMS), to keep better informed about related design activities that might affect the C&TS. These meetings seemed helpful, but it's doubtful that they were an adequate substitute for direct contractor contact.

1.3.2 Documents Received from NASA JSC

There was no shortage of documentation describing the C&TS software and hardware, but most of the documentation was neither current nor coherent. Because of the changing plans for the SSF, the software and hardware design documents lagged the design

process. Versions available during the assessment did not reflect current plans or recent changes to the system design.

Top level documentation such as the Architecture Control Document, Baseline Configuration Document, and Interface Development Document did not reflect the current design. For example, the Space Station Software Standards Document, JSC 30244, referenced by many subordinate documents such as the Contract End Item Specification for the Communications and Tracking System (C&TS), SP-M-002, May 1990, was incomplete. This document was first prepared in September 1986 (version available in the JSC library), and the version at the JSC had not been updated at the time of the assessment. The available version contained statements such as " 6.0 CONFIGURATION MANAGEMENT - To be provided in a subsequent release."

Despite the condition of the documents, they were the only available written record of what the software and hardware were required to do and what was planned for meeting those requirements. More than 110 documents (see Appendix B) were obtained from the JSC for detailed study, and many more documents were scanned and determined not useful. The C&TS is unlike any previous communications system designed for space, so it was necessary to carefully read many of the documents to understand the design concepts. As a result, as much as 50% of the assessment effort was spent on document review.

1.3.3 Other Data Sources

It was initially anticipated that contact could be made with the subcontractor responsible for the C&TS and the subcontractor's subcontractors to obtain their current views on the system hardware and software. Unfortunately, it was learned that there is no C&TS permanent subcontractor presence at the JSC. As a result, it was not possible to learn anything directly from them. There was some indirect information received through "white papers" or written responses to questions from the JSC C&T personnel. Other than this indirect information, SwRI had no technical interchange with the subcontractors regarding topics of this assessment.

Weekly meetings of the C&T technical personnel were held at the JSC. These meetings were attended on a bi-weekly basis by SwRI representatives to learn current thinking, to track program status, and to gain a better understanding of the C&TS. These meetings provided valuable criteria for interpreting and sorting out the incorrect or outdated information from the documentation.

The software language mandated for the system is Ada. There were many information sources used outside of the Space Station program that addressed compilers, coding practices, and benchmarks for the Ada language. In addition, SwRI was in the process of writing Ada code for a large U.S. Army program and had sources of practical information regarding design of Ada source code. This became very important when it was determined that the software language, compiler, and coding techniques had as much to do with memory requirements and throughput as the system requirements and hardware selection.

2.0 PRIMARY ELEMENTS

2.1 Processing Hardware

The processing hardware for the C&TS will be physically distributed over several nodes of the space station. Although the processing hardware will be somewhat standardized for all SSF systems, the actual architecture, processing capabilities, and interconnection will be configured to match the requirements of each Space Station system.

2.1.1 Processor Hardware Architecture

The main components of the C&TS hardware architecture are the Standard Data Processor, the Embedded Controllers, and the Bus architecture.

2.1.1.1 Standard Data Processor

The Standard Data Processor (SDP) (see Figure 1) description is contained in the Configuration Item (CI) Specification for the Standard Data Processor, 152A401-PT1A, May 1990. Further hardware specifications for the Embedded Data Processor (sub processors within the SDP) are available in the Configuration Item Development Specification, MDC H4534, November 1989. Between these two documents and extensive DMS documentation, the characteristics of the SDP hardware are well defined. The Embedded Data Processor is of particular interest because the bulk of the C&T software will reside in a single Embedded Data Processor (plus back up). Key parameters of the Embedded Data Processor are shown in Table 1.

Although the SDP is supposed to be standard for all SSF systems, provisions have been made for more than one version. An SDP version with two Bus Interface Units (BIUs) capable of addressing up to six dual-redundant DOD-STD-1553B buses has been defined, but it is unclear at this date if the C&TS will require three or six local buses. A bus structure has yet to be selected.

2.1.1.2 Embedded Controller

As with the Embedded Data Processor, the Embedded Controller is fairly well described in the documentation (Table 2). Despite the specific information and the statement in the SRU Design Document for the Embedded Controller, 562-SSF-SGVBSP-061, January 1990, that, "The Embedded Controller is designed to be used in all ORUs", the description does not fit all planned controller configurations. For example, per the Software Preliminary Design Document for the Space-to-Ground Baseband Signal Processor, Preliminary, DPB-001, January 1990, the Space to Ground Base Band Signal Processor (SGBSP) is presently designed to use a 80386 processor with different ROM and RAM allocations than the embedded controller of 562-SSF-SGVBSP-061. Conflicting information of this sort made it difficult to correctly interpret the C&TS hardware design documents.

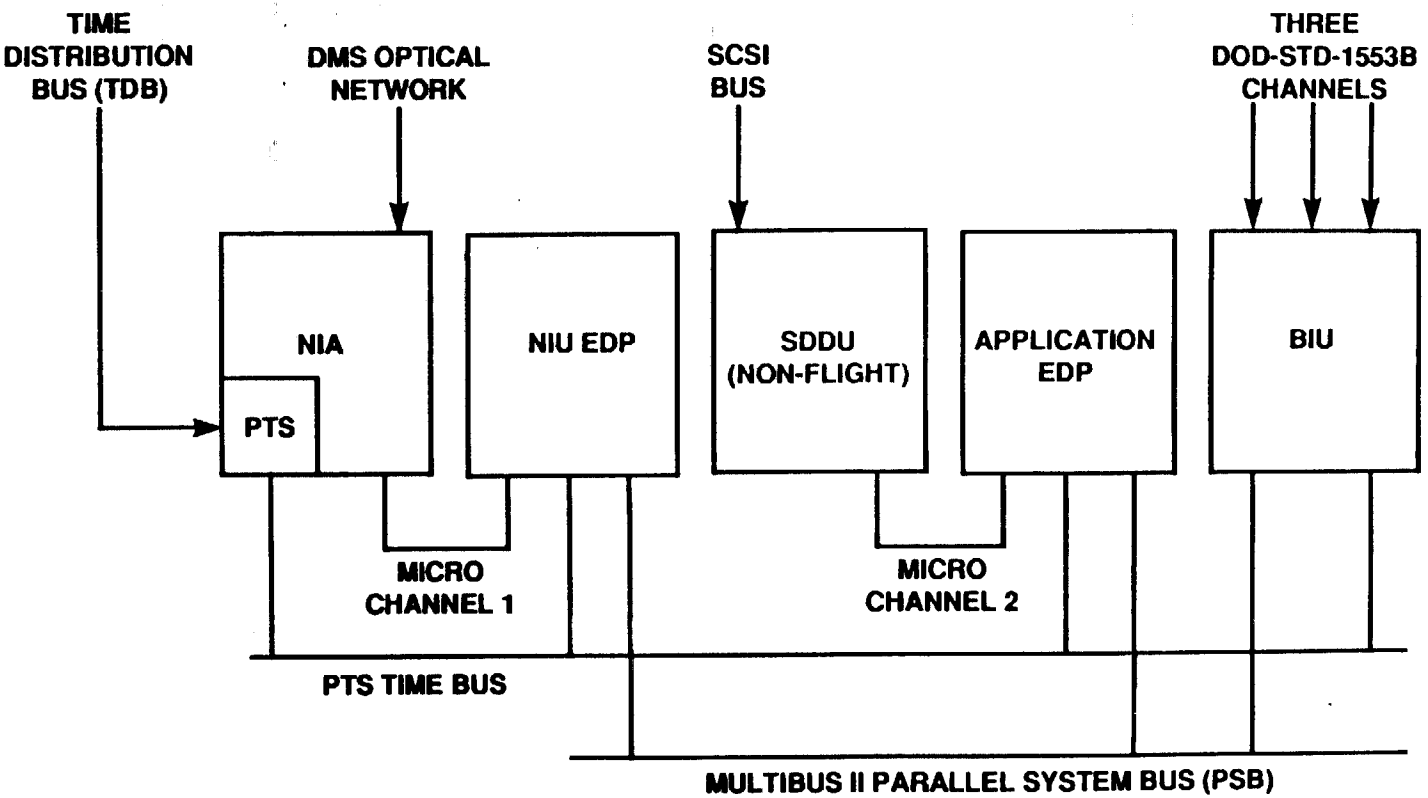


FIGURE 1
SDP BLOCK DIAGRAM

TABLE 1
EMBEDDED DATA PROCESSOR MICROPROCESSOR AND MEMORY

- 32 BIT MICROPROCESSOR (INTEL 80386)
- MATH CO PROCESSOR (INTEL 80387)
- 4 MEGABYTES OF DYNAMIC RAM
  - ADDITIONAL ERROR DETECTION AND CORRECTION MEMORY
  - PROVISIONS FOR MEMORY GROWTH (32 MB)
  - 2 KB NONVOLATILE MEMORY FOR CONFIG TABLES
- EXTERNAL INTERFACES
  - INTEL MULTIBUS II PARALLEL SYSTEM BUS
  - 32 BIT (LEVEL C) IBM MICRO CHANNEL
  - PRECISION TIME SYSTEM
- SOFTWARE COMPATIBLE WITH IBM PERSONAL SYSTEM 2 MODEL 80
- 25 KB BOOT ROM
- SINGLE SRU CONSTRUCTION
- 15 LEVELS OF HARDWARE INTERRUPT
- BUS CONTROL LOGIC FOR MULTIBUS II AND IBM MICROCHANNEL

FROM THE CONFIGURATION ITEM DEVELOPMENT SPECIFICATION FOR THE EMBEDDED DATA PROCESSOR, MDC #4534, NOVEMBER 1989.

TABLE 2
EMBEDDED CONTROLLER

- MICROPROCESSOR AND MEMORY
  - 16 BIT MICROPROCESSOR (INTEL 80186)
  - UV EPROM 128K BYTES DELIVERABLE (256K BYTES INSTALLABLE)
  - STATIC RAM (MICROPROCESSOR) 48K BYTES
  - STATIC RAM (DOD-STD-1553B INTERFACE/SHARED MEMORY WITH MICROPROCESSOR) 16K BYTES
- SYSTEM RESET SIGNAL (POWER-ON OR CMS INITIATED)
- ANALOG SIGNAL MONITOR CAPABILITY (23 CHANNELS MAXIMUM)
  - 8 POWER SUPPLIES
  - 5 TEMPERATURES (INCLUDING 1 SPARE)
  - 1 PRESSURE TRANSDUCER
  - 9 UNASSIGNED
- ANALOG OUTPUT CAPABILITY (3 POSITIVE DC VOLTAGES, 0V TO +10V)
- DIGITAL INPUT/OUTPUT (64 CONFIGURABLE LINES)
- 12 MHz CLOCK OSCILLATOR
- MEMORY (UV EPROM)
- MEMORY (STATIC RAM)

TABLE 2
EMBEDDED CONTROLLER (CONT'D)

- WATCHDOG TIMER
- +5V DROOP DETECTOR
- SYSTEM RESET
- ANALOG DATA ACQUISITION
  - 8 BIT A/D CONVERTER
  - HIGH IMPEDANCE BUFFER
  - +5V PRECISION REFERENCE VOLTAGE
  - 31 CHANNEL ANALOG MULTIPLEXER
  - SIGNAL CONDITIONING CIRCUITS
    - 5 OP-AMP CIRCUITS TO AMPLIFY OUTPUTS OF TEMPERATURE SENSORS (TO BE SUPPLIED)
    - RESISTOR NETWORKS FOR SCALING POWER SUPPLY VOLTAGES TO SATISFY A/D CONVERTER'S 0V TO +5V INPUT SIGNAL REQUIREMENT

FROM SRU DESIGN DOCUMENT FOR THE EMBEDDED CONTROLLER, JANUARY 1990

2.1.1.3 Bus Architecture

The bus architecture chosen for the C&TS is a combination of parallel and serial architectures. A strictly serial bus architecture would pass all information through the processing hardware to the Orbital Replacement Units (ORUs) required to perform the C&T functions (see Figure 2). This is a very flexible architecture, because all data paths are under software control. On the negative side, it raises some reliability concerns, because failure of the processing hardware could disrupt operation of all system ORUs. A parallel bus architecture (see Figure 3) provides less flexibility, because signal paths must be hardwired to specific system ORUs. However, these hardwired signal paths provide access to the signal processing ORUs so that some system capability may remain if control processor failure occurs.

The combination architecture of the C&TS (see Figure 4) provides the flexibility of the series architecture for the high data rate subsystems such as the Space to Ground Subsystem (SGS), and the direct access of parallel architecture for lower data rate subsystems such as the Tracking Subsystem (TKS). In addition, the general purpose CMS bus (DOD-STD-1553B) which forms the "backbone" of the C&TS architecture can be programmed to transfer data or commands. This allows some control of the overall architecture of the system by adding to the local bus and changing the software to accommodate the new ORUs. It is apparent that the bus architecture selected for the C&TS is a very flexible architecture with extensive hardware and software expansion capability.

2.1.2 Processing Capabilities

Once the system hardware elements were defined, the processing capabilities and limitations of the processing hardware were investigated as an aid in determining the most efficient hardware/software distribution.

2.1.2.1 Memory

Memory allocations for the embedded data processors and embedded controllers are described in the appropriate specifications for each ORU (see Appendix B of the Second Monthly Status Report dated September 17, 1990). What is not clear is the rationale for the allocations. Other than the various allocation documents (Processor Resource Allocation Documents), there were no documents found that specifically addressed the memory hardware limit and the criteria for allocating the various amounts of RAM and ROM to the embedded processors or controllers.

There may have been an early C&TS trade study performed to make these decisions, but if so, it was not referenced in any document reviewed. Undoubtedly, there must have been some overall SSF system requirements assessment done early in the SSF program to determine the general requirements for the SDP, but again, no references to it were found. The SDP Embedded Data Processor will be delivered with 4 Megabytes of RAM, but will support up to 32 Megabytes, if necessary. This is a very generous expansion capability, and if used, should be sufficient for any foreseeable increase in

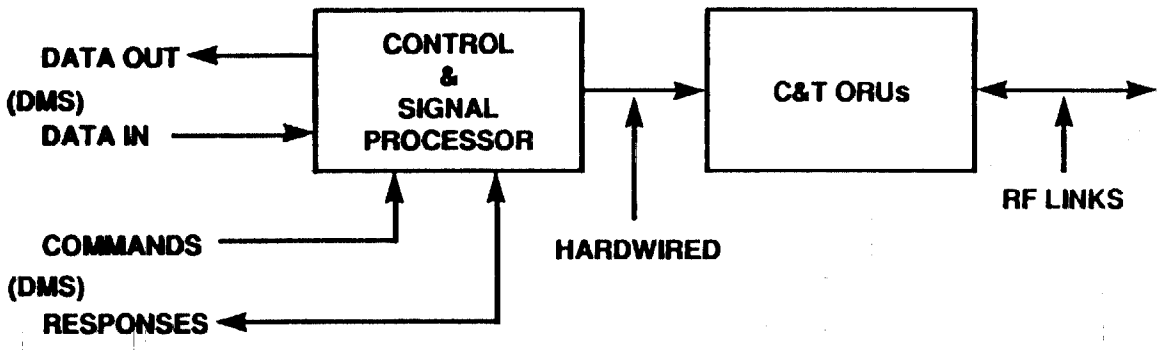


FIGURE 2
COMMUNICATIONS SYSTEM SERIES ARCHITECTURE

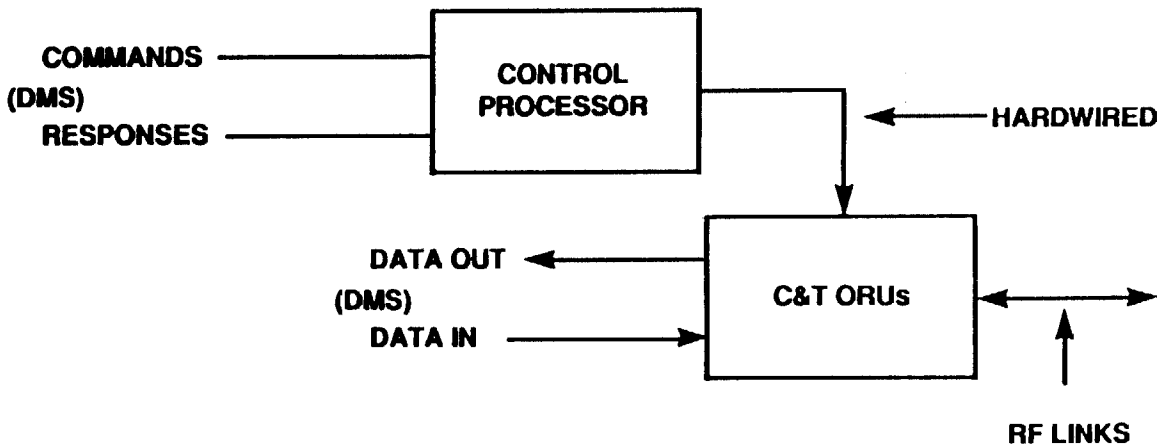


FIGURE 3
COMMUNICATIONS SYSTEM PARALLEL ARCHITECTURE

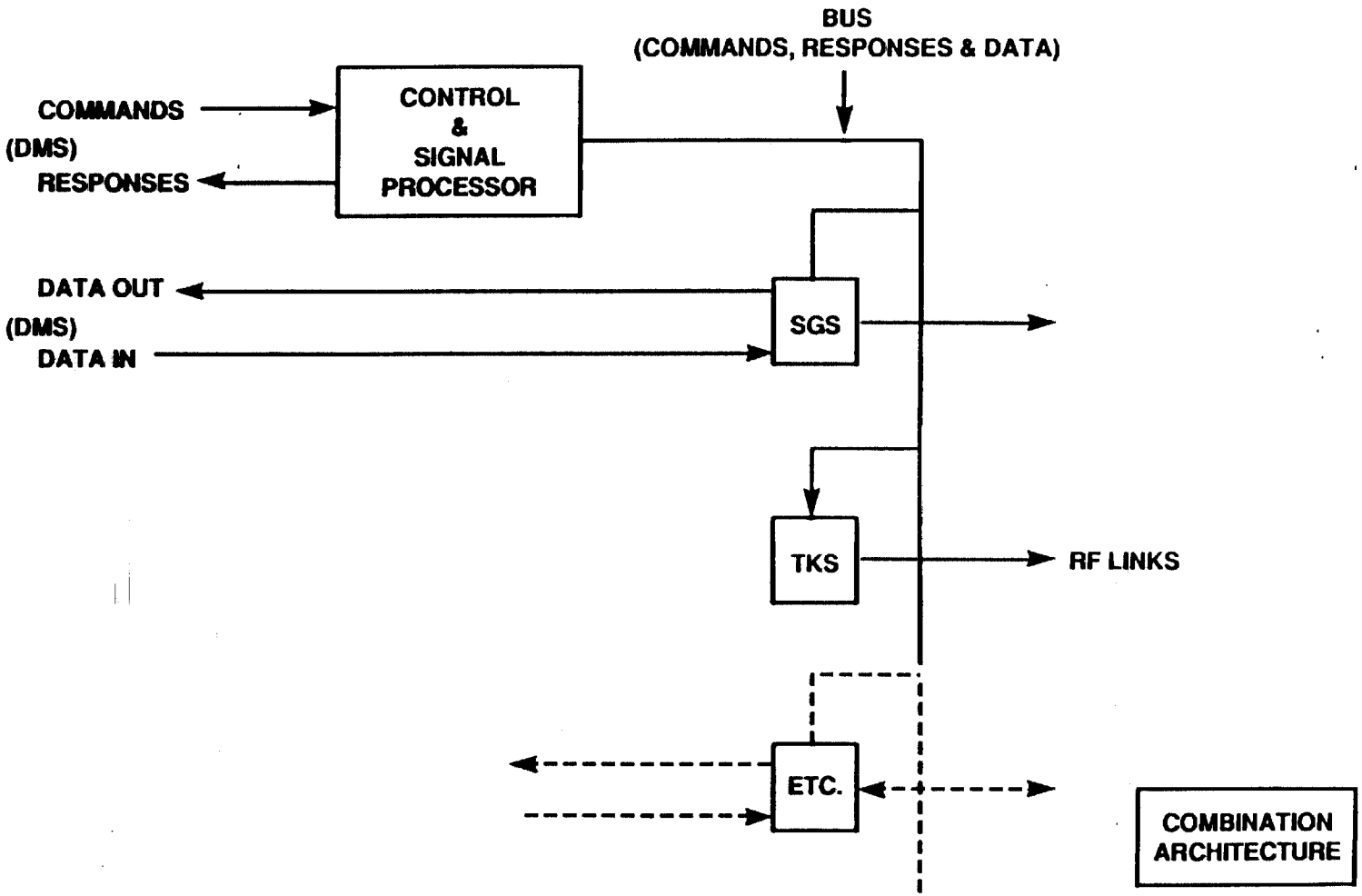


FIGURE 4
COMMUNICATIONS SYSTEM COMBINATION ARCHITECTURE

software requirements. But, the effect of the additional software requirements on throughput cannot be estimated without information on the nature of the increase.

The memory allocation process for the embedded controllers is also unclear. Most embedded controllers have 128 Kilobytes of delivered ROM and 64 Kilobytes of delivered RAM with expansion capability to 256 Kilobytes ROM. No RAM expansion capability is provided. But, depending on the subsystem, embedded controllers have less ROM (64 Kilobytes for SGBSP 80386 Embedded Controller) and some embedded controllers have less RAM (32 Kilobytes for the Video Switches). No mention was made in any reviewed document as to how or why these memory sizes were allocated.

Since multiple subcontractors are involved in the design of ORUs with embedded controllers it would seem reasonable to have some sort of common study or at least a discussion and agreement on the processing hardware to be used. Other than the statement in the SRU Design Document for the Embedded Controller, 562-SSF-SGVBSP-061, January 1990, that "The Embedded Controller is designed to be used in all ORUs", no agreement or study reference was found in the documentation. Yet, as previously noted for the SGBSP controller, not all the controllers are based on the 80186 processors specified in 562-SSF-SGVBSP-061.

2.1.2.2 Throughput

When the assessment was started, it was not known how far the C&TS code development had progressed. After initial review of available software documentation it was apparent that no code had been generated and the detail software specifications had insufficient detail to estimate execution speed and data throughput. Amazingly, the various PRAD releases and the C&TS PDR presentation material presented throughput numbers as if the amount of code and its structure were known to some level of certainty.

The DMS software is reportedly further along in the development process, and has presented some detailed processor loading and latency data to the SSF technical community. Much of this data was described in the DMS Performance Presentation, September 7, 1990. Although the latency and loading data may be accurate, there was no backup material presented to check the derivation of the data.

The DMS report presented many tables with exact timing values for each DMS metric and many compilation diagrams and tables showing how these values add up. In some cases models were referenced, for example the Nyles Heise/IBM Dynamic Model, but no description was presented explaining how these models were used to generate the timing data. Despite all the detailed data, the report did not describe the analysis process. General statements, such as "Several methods have been used to make DMS performance projections, depending on the maturity of the component involved," are all that was provided. Without the derivation information, it was not possible to assess the validity of the data.

In reference to the local bus performance for the system applications, the same DMS report stated, "Local Bus I/O modeling is still being developed, and (performance) is

dependent on completion of the detailed software design." Apparently, the DMS software personnel also believe that the software must be developed to some extent before representative performance estimates can be made.

The lack of adequate software descriptions for many of the subsystems and the complete absence of any software description for other subsystems such as the UHF and Tracking Subsystems resulted in no meaningful throughput estimates for the C&TS. The most that could be said about throughput was that there are too many interrelated factors that remain undefined at this time to make any useful estimate of overall system throughput for either information or control.

2.1.2.3 Input/Output (I/O)

Inputs and outputs for each of the processing elements, i.e. SDP and Embedded Controllers, are fairly well described in the specifications for each. The ORU inputs and outputs are not as well described. Some subsystems such as the Space To Ground Subsystem have ORU descriptions that include details of each I/O situation. Other subsystems, specifically the UHF and Tracking subsystems, do not contain well defined hardware descriptions. Also, the overall local bus interconnection situation has not been resolved leaving questions about the required number of ORU connections for each local bus. The C&TS PDR presented several local bus scenarios all supporting 76 ORUs (see Figure 5).

2.1.3 CMS Bus Architecture

The overall analysis of the CMS bus architecture and its capabilities, including the efficient communication between the CMS and the ORUs, is a major area of concern in the implementation of the C&TS. The physical layout of SSF equipment dictates the location of the ORUs that control functions such as antenna pointing, activation of on-line systems, activation of spare systems, etc. The dynamic nature of the changing SSF structure has caused the CMS architecture to be subject to significant revisions. Without knowing the numbers and locations of the equipment components in the CMS architecture, it is not possible to establish the optimum architecture of the CMS.

Eventually, however, the specific number and the locations for the ORUs will be determined. Further, Standard Data Processor (SDP) designs have been determined. Further, the DOD-STD-1553B (Aircraft Internal Time Division Command/Response Multiplex) Data Bus standard has been chosen as the method of interconnection. Therefore, the bus throughput, rules of engagement for the bus controller, SDP and ORU processing capacities, and many other items are relatively fixed. While it will be no small effort to achieve this goal, the best arrangement for the CMS architecture can be reached once the number and location of the ORUs is finally determined.

Understanding this very important problem exists, a systems engineering and integration trade study on the CMS architecture for phased assembly sequence was completed and published on 31 May 1990 (SY-01.3-015). Because the results of the trade study are critical to the design of the CMS architecture, it was carefully reviewed. The

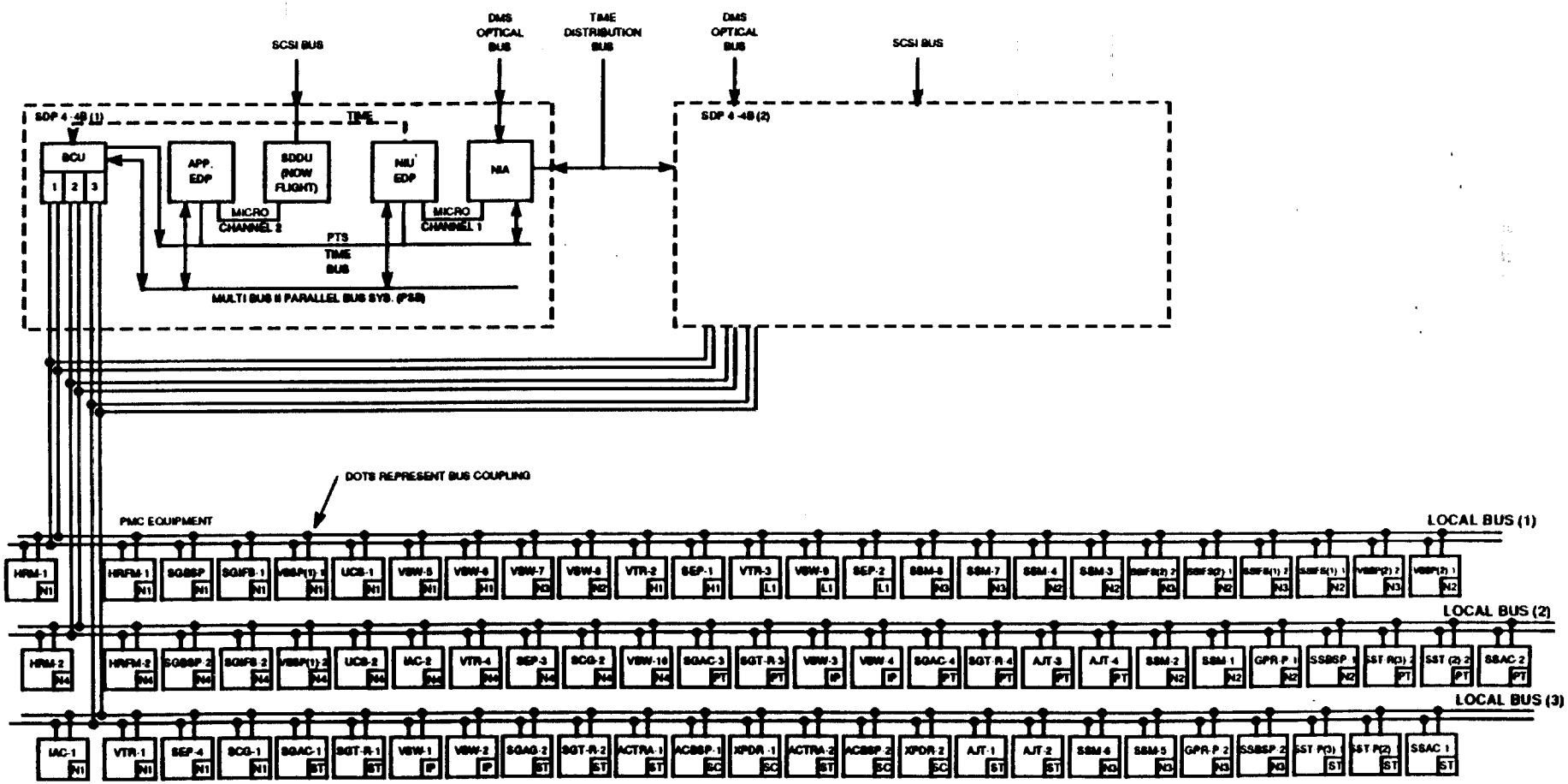


FIGURE 5
PDR POSSIBLE CONTROL AND MONITOR SUBSYSTEM LOCAL BUS INTERCONNECTION (ASSEMBLY COMPLETE) DIAGRAM

apparent objective of the trade study was to determine the best CMS architecture based on the SSF structure at the time of the initiation of the report. The insight from running the trade study based on what was thought at the time to be the SSF structure was deemed worth the cost. The stated objective of the trade study was "to determine the optimal CMS architecture for the phased assembly sequence; permanently manned configuration (PMC) to assembly complete (AC)".

However, in the findings of the trade study the actual objective accomplished is stated to be quite different. That is, the trade study findings restate the objective to say, "to compare the relative merits of the five architectures." The difference between the stated initial objective and the accomplished final objective is a serious problem.

First, the optimal CMS architecture for the selected SSF configuration, or any future configuration, was not retrieved as the result of the trade study. Instead it was determined that none of the five CMS architectures that were being considered could have been successfully applied because of various failings. Choosing the best of five failing CMS architectures cannot be considered equivalent to determining the optimal architecture.

Second, because of the engineering approach, updating the trade study for new SSF configurations is at least as difficult as doing the original trade study. That is, the trade study did not present a method to analyze new CMS architecture each time a new version of the number of ORUs and interconnecting architectures is determined. At each point such as this, following the trade study strategy, it is necessary to redo the trade study for each revision of the CMS architecture.

It is assumed that this process would continue, using error information gained from each failed architecture, until a working CMS architecture is attained. This is obviously an unworkable plan. This can be seen in the actual events since the trade study has been released. Because of the dynamic nature of the space station development, revisions of the CMS architecture are occurring quite regularly, but the trade study analysis has not been repeated.

Then again, the overall engineering approach was not the only problem discovered with the trade study. The implementation details of the analysis contained fundamental logic flaws, data inaccuracies, and incorrect assumptions. Therefore, it is not recommended that this particular analysis be repeated.

But yet, an analysis of the CMS architecture is necessary, and the original objective is still warranted. A later section discusses the recommended approach to select the optimal CMS architecture. However, even if the overall engineering approach were changed, it is important to know the detail problems within the current trade study so that they would not be repeated in the future.

The items presented in this section are the problems that were discovered with the trade study. There may be more problems, however these stated are sufficient to invalidate the methods and findings presented in the trade study. Naturally, some of the findings are more important than others, but they are not presented in their order of importance.

Instead, they will be presented in the order of the sequence they occur within the trade study. This will make it easier to follow each of the described problems through the document including its attachments. Each discussion will take the item as it was presented, however some of the details associated with the items appear in later sections of the trade study, as tables, raw data, or additional explanations. These details will be brought forward and discussed with the initial description to make the comments in this research complete within a single section. Whenever the raw data and the method of computation were known, a validation computation was completed. When a computational error in the trade study is pointed out, the data of the trade study was used unless it is specifically pointed out differently.

2.1.3.1 Objective and Requirements

This section expands on the earlier discussion concerning how the basic engineering objective gets diverted from the stated objective. The trade study paragraph 3.1 states the primary objective as follows:

> "The primary objective of the study was to determine the optimal CMS architecture of the phased assembly sequence; permanently manned configuration (PMC) to assembly complete (AC)."

This is a required, desired, worthy, and attainable objective. The method to meet the objective is in this same paragraph, and reads:

> "The optimal architecture was selected by evaluating quantitative data from selected criteria and utilizing a scoring and weighting system to select an architecture with the highest score."

The basic method statement is not flawed. The optimum architecture could be determined as a product of a criteria scoring and weighting system. But instead of yielding an optimum architecture, the criteria was applied to a set of architectures, which were then compared relatively to each other. If the set included every possible CMS architecture, then it is feasible that the resulting choice would truly be the optimal CMS architecture. But of course, evaluating all possible architectures is not a practical scheme (although it is inherent in the trade study strategy). If, by chance, the optimal architecture was in the first set of five possible architectures chosen for evaluation, then of course it is again feasible that it could be found. But, since none of the architectures met all of the criteria, it can be stated that the optimal architecture was not discovered.

If one or more of the architectures compared did meet all of the criteria, then one would have a working system, at least for that version of SSF. But considering the continual nature of the system transformations, it is obvious that configuration revisions of the SSF will continue. Repeating the trade study's process would be expensive, and would result in little or no assurance of attaining a workable CMS architecture, let alone the optimal CMS architecture.

Notice that it is possible to have a "correct" output from the search for optimal architecture analysis to show that given the number of ORUs required, the number of bus lines allowed, the SDP processing capabilities, the weight and power budget, etc., that there is no possible way to interconnect the C&TS equipment together and still meet all of the criteria. Assuming the criteria were chosen correctly, this would indicate a redesign of the ORUs or SDP, reconsideration of the number of SDPs (since it was constrained), reconsideration of the number of bus lines possible (if it was constrained), increased power or weight budgets, or a host of other possible corrective actions. In this situation, a thorough analysis would indicate possible corrective actions to the system components to develop the optimal CMS architecture.

2.1.3.2 Scoring

The scoring system used was "... chosen such that the best possible score is four (4) and the worst possible score is zero (0) with the exception being criteria with calculated utilization of over 100 percent which will receive a negative score. The weighting system was chosen such that the most important criteria receives a weighting of five (5) and the least important criteria receives a weighting of one (1)." This system was an attempt at combining all the information into a single number total score which could then be used to signify the optimum architecture.

This is a questionable approach with much potential for error. The problem is the concept that a configuration which fails to meet one criterion, but receives high scores for all other criteria will be determined a better choice than a configuration which meets all criteria. The single number total score concept is inappropriate for such a different set of criteria associated with the SSF CMS architecture. The scoring methods used to determine the individual scores needs to be understood before the criteria judgments can be detailed properly. The first scoring scheme applied in the trade study, listed in paragraph 3.3.2.1, is the "percentage" scoring mechanism that is used in four of the nine criteria cases, i.e., the Processor Loading, Memory Utilization, Bus Loading and Used RT Ports. The score is equal to the percentage (based on the case amount compared to some standard) which is then multiplied by negative four and divided by 100 percent. This number is then added to four to get the unweighted score. This allows a range of scores from plus four to less than zero. The negative numbers result when the data shows that the case requires more of something than is available. According to the scheme, the negative numbers can be balanced by high values in the other criteria areas. This is in contrast to the fact that the negative number indicates the case fails the specification. Allowing a negative number is a bad practice, because a failure should stop the evaluation. Because this is a physical real system that must operate, a failure cannot be balanced out by the higher scores from the other factors.

Also noted within the percentage scoring method, is that multiple occurrences of a criteria (for example, the cases with more than one processor) occurrence requiring the most resources is used to calculate the score. This logic leads to further obscure the conclusions. For example, a case with multiple buses will usually have one bus that is more heavily used than the others. It may be that there are six buses, with the other five buses lightly loaded with the backup units assigned to them. Nevertheless, only a single

number, based on that worst case, is brought forward to characterize the architecture. Another case may have only four rather heavily used buses (which could be altered significantly to the negative side when backup ORUs are activated on the bus), but none actually exceeds the worse case in the previous six bus example, but all four exceed the other five. A single bus score will be included into the total that indicates that the second case is preferred, which is clearly not true.

The second scoring scheme used with the power, weight, and volume criteria is related to the amount of resources needed. The score is computed by assigning a score based on a linear curve between four, meaning no resource is needed, to zero assigned to the maximum value obtained for the criteria compared to all of the architecture cases. There are two problems with this scheme.

First, the worst score of zero is given to an architecture case that may fall within the resource budget, but is still heavily penalized because it has the maximum value in that category. Also notice that according to this trade study method, there will always be a score of zero assigned. Second, the best case score will be assigned to an architecture case that may be extremely deficient in its capability to meet the mission performance needs. Since two of these criteria have been assigned the maximum scale weight of five, these factors become more important overall than the mission performance. As the individual criterion are discussed that use this scale, these situations will be illustrated with examples.

2.1.3.3 Selection Criteria

The trade study paragraph 3.3, and its subparagraphs, present all of the selection criteria. It is pointed out that each criterion is "weighted (numerically or by pass/fail) to relative importance in determining the effectiveness of alternatives." There are two problems with this technique as used in this system of criteria comparisons.

First, all of the items should have had pass/fail thresholds that stop the evaluation. That is, if the CMS architecture processing requirement of a particular case being considered exceeds the amount budgeted, the architecture case should have failed the criteria and no longer been a possible candidate. In fact, contrary to this thinking, it is stated in the conclusion that even those few pass/fail items that the trade study did include, were not factored into the scoring system. This rationale was followed because the trade study engineering approach was based on choosing the best possible out of a set, so the process would not stop, rejecting the system as a possible architecture as it should have. Instead it continued on, assuming that it may score higher than another architecture because this one takes less power, or is lighter. This would make it the best overall architecture, even though it wouldn't actually have the processing power needed.

Second, the weighting of the criteria is so biased toward the processing loading, the power, and the weight, that the other factors could have been left out of the trade study without any change in the result. Hence the trade study's conclusion, "the power and weight considerations versus available processor utilization are key drivers for selection of a recommended architecture" must be considered a trivial solution. That is, none of the

work completed in the trade study influenced the conclusion after these initial assumptions were established.

On the contrary to this situation, the other criteria should have been factors in the final conclusion. In fact, paragraph 3.3 states the selection criteria are driven from program requirements, military standards and requirements in the SOW. A cross reference matrix to these would have been a desireable attachment to show why such exaggerated biases in the weightings were employed based on these requirements.

2.1.3.3.1 Processing Loading

The processing loading has a scaled weight of a five (the maximum assigned), and was "based upon expected frequency of required functions and estimated lines of code for each function plus processing to service messages/transactions between LCA and CIs (as applicable) analyzed for each processor." It was computed using a formula found in paragraph 3.3.1.3 of the trade study.

The trade study assumed a maximum processor MIPS allocation based on 3.1 MIPS per SDP with 1.581 MIPS being the allocation for the CSCIs at PMC and 1.581 MIPS at AC. This data came from the Dec 89 PRAD, and the percent utilization was based on the contractor's allocation at that time. Because the trade study was published on 31 May 1990, it did not include the updated processing values released in the May 1990 PRAD. These same allocations were lowered to 1.331 MIPS in the resubmittal, or only 84 percent of the value used in the trade study. It is understandable that the values are estimates and will change, but because of a lack of currency and the rigid engineering approach, this data, as well as much of the later data, was known to be invalid before the trade study report was ever published and nothing could be done to correct the error short of reaccomplishing the trade study. It is very apparent that the static input method used in the trade study was inappropriate for the system being measured.

The equation employed to estimate the use of the processor was designed to take into account both the fixed processing activity of the CPA and LCA load scenarios, assigned as .291 MIPS and .4 MIPS respectively, and the processing loading due to the message transactions of the BIUs. It was not clear where these scenario estimates were derived but it is apparent from the terminology that these values are based on machine language instructions. On the other hand, values of 400 and 200 Source Lines Of Code (SLOC) per message were used in the equation to account for the message loading factor. The estimate of SLOC is based on ADA code usage to accomplish the task. These SLOC factors, after being multiplied by the number of messages, were incorrectly added to the .291 MIPS and .4 MIPS machine language instruction scenario values. A trade study assumption established a factor of 4 machine language instructions per SLOC, but failed to recognize it was a needed conversion to be used at this point. Hence the values for processing loading determined from the equations are obtained by adding unlike quantities, making the final computed data incorrect even if the estimates were not.

2.1.3.3.2 Power

The power criterion was also weighted as a five. It was to be the sum of all the power requirements for all CMS equipment excluding the C&TS ORUs on the 1553B data buses as a note in the trade study indicated. Paragraph 3.3.1.5 in the trade study added some information that the equipment normally powered off was not included. The power data for the SDPs and RCs were obtained from the DMS critical item specifications.

The score established for best power usage situation of zero Watts is given a four. At the other end of the scale, the power consumed in the worst case has a value of zero. Naturally power is a critical resource on SSF, however using this criterion, having no CMS equipment yields the highest score, and using extensive equipment yields the worst but still passing score of zero. In fact the power usage should follow a system that shows that if the power allocation is not exceeded, then the architecture should qualify as a possible architecture. Further, given everything else equal, the lowest power usage would be associated with the optimum choice. The fact that backup power requirements are not considered is of concern. It must have been assumed that the failed equipment has been powered off, which is not always true.

2.1.3.3.3 Weight

The weight criterion also has weight scale of a five and uses the same scoring method as was used in the power criterion. The value was to reflect the sum of all the weights for all CMS equipment. The cable weight was obtained from the DMS "Local Bus Trade Study" by Landen and Nossaman. The backup equipment must have been included since it was not stated as being excluded in this case. Once again, no consideration was given to being under or above a budgeted weight amount.

The score established for best weight usage situation of zero pounds is a four. At the other end of the scale, the weight in the worst case has a value of zero. Once again, this system defines no CMS equipment as the optimum, giving it the highest score, and using extensive equipment yields the worst (but still passing) score of zero. As in the power case, the weight usage should follow a scoring system that shows that if the weight allocation is not exceeded, then the architecture should qualify as a possible architecture. Given everything else equal, then the lowest weight usage would be associated with the optimum choice.

2.1.3.3.4 Memory Utilization

The memory utilization used the percent scoring system and was scaled with a weight of three. The source used for the memory allocation basis was the Dec 89 PRAD. The source for the memory used data is unknown, but it did not seem to use the SDP memory estimate contained in the Dec 89 PRAD. The estimated data used was provided in paragraph 3.3.1.4 of the trade study, but it cannot be validated. Since the method used to determine the amount of memory utilization is the percentage method, the comments including the single worst case comment is also applicable for memory utilization. The worst error in applying this measuring system is that it can not be considered correct to

penalize an architecture case by giving it a zero worth, because it uses all of the memory space assigned to it. It is just as wrong to assign a negative number to an architecture that exceeds its allocation. Further, it seems strange to give the highest, or optimum score, to a system that uses no memory allocated to it.

Note that it appears, but it is not documented, that the code memory space utilization estimates were based only on static storage value and not on runtime memory usage. The overall problem of estimating storage use is not a simple one and it is addressed more thoroughly in the later section under compiler choices. The details of general storage space problems will be deferred until then. The point is that the estimates of storage space are as unsubstantial as the system used to compare the data values.

2.1.3.3.5 Bus Loading

The bus loading utilization uses the percent scoring system and was scaled with a weight of three. The percentage basis used for the bus capacity maximum allowed was the bus data rate of 1 Mbps. The I/O channel allocation standard in the Dec 89 PRAD of 700 kilobytes per second was not used. The source for the message used data was provided in the appendix to the trade study. Since the I/O utilization is the percentage method, the comments including the single worst case comment is also applicable for memory utilization. Once again note that, in applying this measuring system an architecture is considered of no worth by giving it a zero score when it uses the I/O capacity allocated to it, and the highest, or optimum score, is given to a system that uses no I/O.

One error that is associated with the 1553B Bus I/O utilization is that the data rate of 1 Mbps should not be considered the maximum throughput (it is much better than the 700 kilobytes noted above, but still wrong). There are several factors that limit the throughput achieving its line speed that were not incorporated into the trade study. The overhead associated with the 1553B standard requires a minimum of a 4 microsecond delay as an intermessage gap. There is also a 4 to 12 micro second response time window allowed for a valid command word. Neither of these 1553B bus overhead components are included in the trade study computations. However, these bits can amount to 8 to 12 bits every message. The short control message exchange can be as little as 40 bits, hence the 8 to 12 bits overhead (20 to 30 percent more time per message), is a significant addition to the bus loading computations that was ignored when a 1 Mbps value was used for the I/O capacity of the bus.

The average bus loading due to aperiodic messages is based on one tenth of the message length per second. That is, if a message is 700 bits, the loading computed for it is 70 bits per second. The basis for this assumption is not clear.

The bus loading data does not include message traffic associated with the backup equipment on the buses. The worst case loading for any single bus is when all of the equipment attached to that bus, including the backup equipment, is operating. Using the data in the trade study the bus loading was the smallest problem area. However, the procedure to consider bus loading data only for the primary operational mode, leaves the

unanswered question, "If the backup ORUs on a bus must be activated, will the bus still support the combined message loading?"

And finally, in certain cases, the bit count for the messages that was used was exactly 20 bits (one 1553B word) higher than it should be. This is true for periodic transmit and the aperiodic transmit messages such as VS7M being a 8 bps load versus a 6 bps load for VS1C. There appears to be something besides the Command Word, the Status Word, and the Data Words included, but it was not defined nor could it be determined.

2.1.3.3.6 Volume

The volume criterion was assigned a weighting factor of two. Apparently the space problem on the shuttle is 2.5 times less significant than the weight problem. The same problems that applied to the power and weight evaluation methods apply to the volume criterion, except that additionally a computational error in the case 5 PMC data of the weighted score of .11 units. The significance of this error is realized when it is seen that the weighted value for case 4 Assembly Complete was .1. This indicates that the error in case 5 was more than the total value computed for case 4. Also, the cases with the worst volume use is assigned the value of zero, indicating that an .11 error is significant in that it is more than these scores.

2.1.3.3.7 Spare RT Ports

The Spare RT Ports criterion was assigned the lowest weight, i.e., one. It used the percent scoring system, and considered the worst situation to be when all 31 ports available for use are being utilized, and the best case when a bus has no ports assigned for use.

The unused spares is a very misleading parameter. The first problem is again associated with the percent scoring system in that the worst case bus presented for each architecture is the only one that is included in the comparison between architectures. This method hides a much larger spread between buses than is indicated by the data. Also, note that any expansion requires the need for the ORU to be there and be functional. This demands processing for the bus activity, message processing for the ORU's traffic, increased weight and power consumption, and a host of other variables. It is not possible to simply expand the bus because the address limit of the 1553B bus (31 nodes) has not been reached. Therefore, it is not appropriate to favor one bus over another because the ceiling for the address availability in the 1553B scheme has been approached in that bus. In fact, contrary to that logic, the Contract End Item indicates in para 3.3.9.1 that growth allows for replacing ORUs with new technology, but does not indicate additional ORUs would ever be added.

2.1.3.3.8 1553B Bus Impedance

The Bus impedance criteria had a pass/fail weighting and was a nodal analysis of the 1553B bus impedance as viewed from each remote terminal on each bus to determine

if degradation of wave forms violated the DOD-STD-1553B requirements. The bus impedance analysis tool used was the SCI Technology, Inc. simulation program. There was a comment in the trade study general findings that indicated recognition of simulation problems in the results. It is as follows:

"At this point in time it is not known whether one or both of the models are inaccurate."

The raw data used, and the outputs of the simulation were provided in the appendices. The trade study assumed that the C&TS bus routing would minimize the stub lengths at the expense of added bus length. There is a basic flaw in that assumption. Extending the bus to go to each node increases the exposure of the bus, therefore increasing the probability of bus damage and total bus failure. It is more logical to increase the exposure for the single node, rather than to increase the exposure of the entire bus. Further, adding bus length increases weight and volume factors if the bus is looped out and back to a single node. This seems to be an unusual choice, because up to a 20 foot stub is acceptable, and used in some parts of the structure, and even longer stubs are not ruled out by the 1553B standard.

It was stated that the bus impedance criteria was not included into the scoring system, however a lengthy discussion was available which presented the tables and scores obtained from the simulation data. The trade study claimed that the simulated waveforms indicated failures of the zero crossover caused potential problems with all five architectures. The trade study indicated that the simulated waveform failures for individual bus structures occurred from a minimum of 13% of the nodes on the local bus 4, case 1 showing failures to a maximum of 50% of the nodes on local bus 2, case 1 failing. The average node failure for all of the nodes simulated was 34 percent. Not one bus structure simulated was reported to have successfully met the zero crossover specification. Additionally, the case 2 architecture was reported to have come very close to not meeting the minimum signal amplitude specification.

If this outcome were correct, it would be a disastrous result. Obviously, all of the buses would have been ruled out as possible CMS architectures because of the zero crossover specification and the trade study would have had to stop. Oddly, that was not the reasoning used, and for some unknown reason, the conclusion was that only case 2 needed to be eliminated because of impedance/layout concerns. Perhaps it was ruled out because it was also claimed to have almost not met the minimum signal amplitude specification.

In fact, all of the conclusions, tables, and logic with the bus impedance findings are in error. Analyzing the SCI Technology Inc. data in the trade study's appendix reveals that every node, on every bus, passed every simulation.

Also, the lowest voltage input in case 2 for the most distant ORU is still at 2.0 volts, which is well above the standard minimum of .86 volts. Note also that the output value at the source is only 4 volts, indicating a relatively small drop between source output to ORU input. The maximum the standard requires at the output is 18.0 to 27.0 volts

loaded at 70 ohms. The source output value used for the simulation is unknown, but it appears that it could be boosted to improve the ORU inputs if a real problem did arise.

2.1.3.3.9 SDP Availability Risk

The SDP Availability Risk is associated with the different architectures being evaluated needing SDP types not currently supported by DMS. This follows the pass/fail rule that needs to be evaluated for each case. Since the criteria that used pass/fail rule were not included in the scoring system, it is assumed that this criterion will be relegated to a discussion just as the bus impedance criterion was done earlier. Contrary to these expectations, the only comment related to this criterion discovered in the trade study was found on page 35 related to the case 5 findings. It stated the following:

"This architecture requires an SDP with supports two BIUs."

Assuming this sentence should read "which" instead of "with", it still does not fail case 5, nor does it pass case 5. In fact, it is not certain what impact it was intended to have on case 5, because it made no judgement in relation to this statement. Since this is the only place the trade study makes any comment about SDP availability risk, it is not at all clear why this criterion was included.

2.1.3.4 CMS Architecture Trade Study Conclusions

There are many assumptions that needed to be made to complete the trade study, but those chosen turned out to be incorrect and misleading. The trade study justified some assumptions as correct by generally referring to requirements documents or other studies, but the rationale for most of the assumptions were not made available for review. This is especially true in cases where the assumptions seemed unreasonable. The engineering methods of criteria evaluation, the scaling procedures, the findings and conclusions of the trade study are all unacceptable.

The recommended approach to do such a trade study is that the criteria should be established and become inputs to a system of equations and interrelated maximums and minimums. This would be accomplished in a manner similar to the linear programming conceptual approach at selecting the optimal solution given an objective equation and the set of constraints. The key is that for each version of SSF, a known set of factors will immediately result, and from that point an analysis could indicate the optimum architecture, or determine that if one does not exist, to recommend various solutions to achieve one.

This engineering solution approach is intended to be as dynamic as the subject question. That is, as events drive new or different SSF structures, these factors can be input into the system, and an optimum CMS architecture, or specification conflict, will result. This is a specific engineering design tool which supports the specific design process of the CMS architecture.

The details of the trade study errors were not presented here to prove that this trade study was inaccurate and contained numerous logic flaws. The trade study report

recognized this itself when it qualified the results by saying that architecture 3 was the "recommended architecture of the study" and not the recommended architecture for SSF. The details of the study were presented as a caution that any decision based on information originating from the trade study should be used with great caution. This is a very serious situation if hardware design decisions are actually based on the results of this trade study.

2.2 Software/firmware

The other half of the C&TS is the software/firmware. The Control and Monitor Subsystem (CMS), ORU Firmware, and Software Language were considered the primary assessment areas for software.

2.2.1 Control & Monitor Subsystem

At the heart of the Control & Monitor Subsystem (CMS) is a Standard Data Processor (SDP). An overview of the software content of an SDP is presented in Figure 6. The SDP contains two Embedded Data Processors (EDP) as described in section 2.1.1.1. As depicted in Figure 6, one EDP contains the Network Operating System (NOS) while the other contains the system applications, the Data Management Applications (DMA), and the Runtime Object Data Base (RODB). DMA includes Operating System Ada Run Time Environment (OS/ADA RTE), Data Storage and Retrieval (DSAR), Standard Services (STSV), User Support Environment (USE), and System Management (SM).

The following sections describe the CMS application software, the DMA software, and the RODB.

2.2.1.1 Control and Monitor Application Software

There is a significant delay between the time a change is agreed upon until the change is incorporated in the C&TS documentation. Although an architecture change has reduced the CMS application software to a single CSCI, all Space Station Freedom (SSF) documentation reflects the development of two CMS CSCIs, each residing in a separate SDP. The two applications are the Local Controller Application (LCA) and the Control & Monitor Processor Application (CPA).

The purpose of the LCA is to control the operation of the C&TS equipment, monitor the health and status of the C&TS equipment, monitor data received from the C&TS equipment, detect faults in C&TS equipment, perform fault isolation and recovery, communicate with the CPA, and maintain files on the Mass Storage Unit (MSU). Although not consistent with the release chronology of the Software Requirements Specification (SRS), the Software Preliminary Design Document (SPDD), and the material presented at the Preliminary Design Review (PDR), it is clear that the subfunction decomposition presented in the SPDD (SY-33-002 5/90) represents the software breakdown. The subfunction decomposition presented in the SPDD is reproduced as Figure 7.

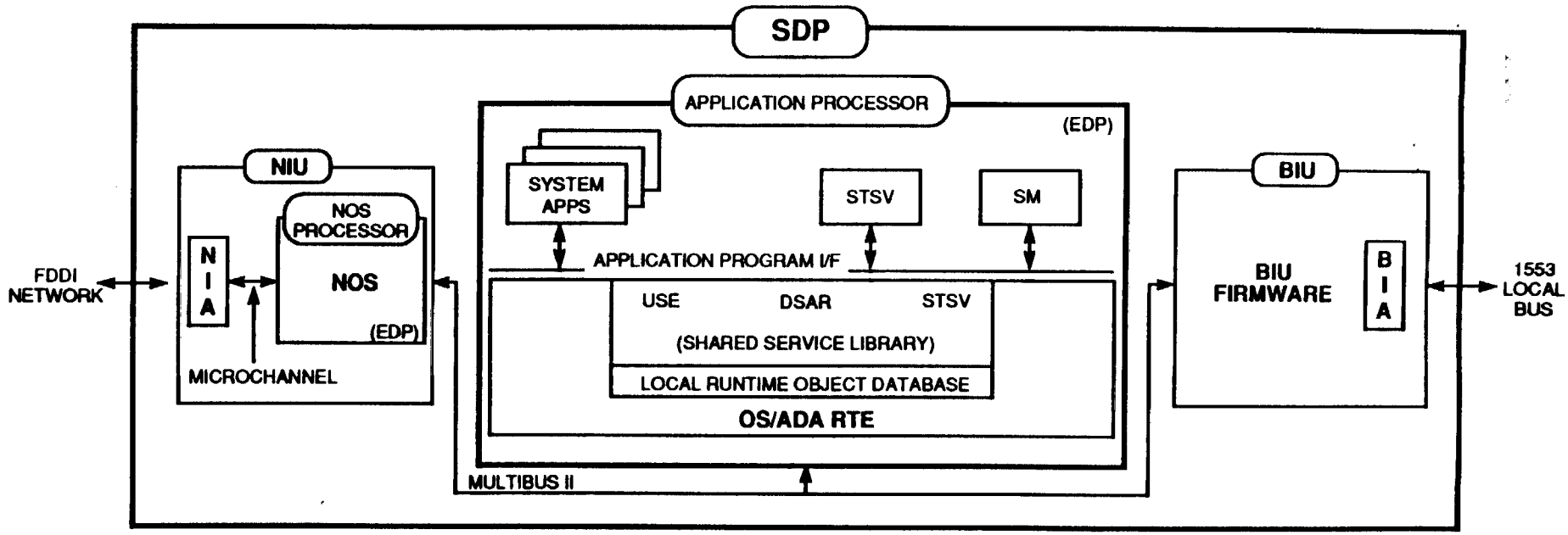


FIGURE 6
STANDARD DATA PROCESSOR ORGANIZATION

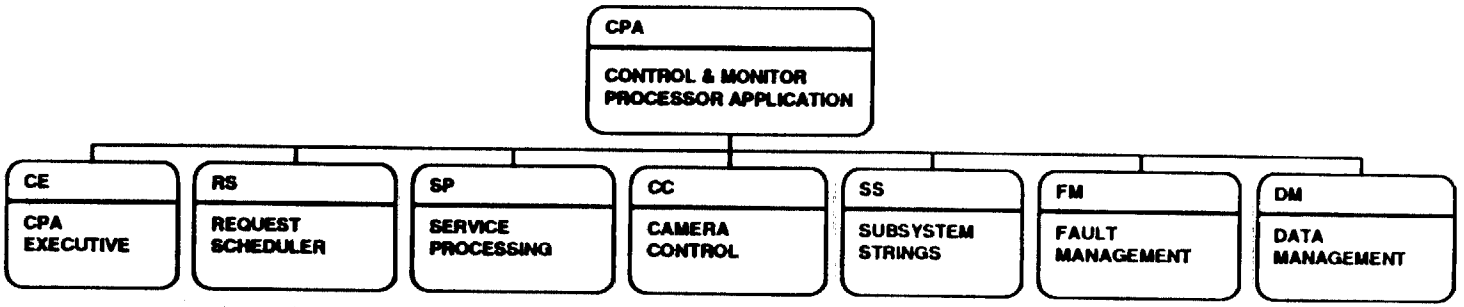


FIGURE 7
CPA SUBFUNCTION DECOMPOSITION FROM THE CPA SPDD

The purpose of the CPA is to initialize the CMS software and C&TS equipment configuration, control the operation of the C&TS equipment, supervise C&TS equipment monitoring and reporting, establish C&TS fault isolation and recovery processing, interface with the LCA, and maintain files on the MSU. As with the LCA, it is clear that the subfunction decomposition presented in the SPDD (SY-33-001A 5/90) represents the documented software breakdown. The subfunction decomposition presented in the SPDD is reproduced as Figure 8.

While no official documentation for the combination of the two CSCI's into a single CSCI exists, SwRI has projected the subfunction decomposition of the combined CSCI as illustrated in Figure 9. This Control and Monitor Application (CMA) breakdown combines the two executive subfunctions into a single subfunction and combines the LCA Database Services and the CPA Data Management subfunctions into a single subfunction. All of the other subfunctions from LCA and CPA appear in the combined CSCI subfunction decomposition.

The latest Processor Resource Allocation Document (PRAD) dated May 1990 indicates that the memory usage in the CMS SDP is 40 percent over the available memory. However, this estimate was obtained by adding the memory requirements of the LCA and CPA CSCIs, and with the exception of placing the two CSCIs in the same SDP, did not account for recent changes which have not filtered into the SSF documentation.

By comparison, the estimated SDP memory requirement for the Data Management Application (DMA) as presented in the PRAD 12/89, PRAD 5/90, and DMS Processor Resource Allocation Manual (PRAM) 7/90, has steadily declined as DMA software development has proceeded. Although the PRAD 12/89 contains numerous mathematical errors which suggest cautious use of the numbers provided, the steady downward trend in the DMA memory requirement is clear.

The SwRI CMS application estimate, presented in Table 3 is based on reduction due to architecture changes, to use of standard services software (see section 2.2.1.2), and to use of common code, e.g. common subroutines. The architecture changes include combination of the two original CSCIs and reduction in system hardware, e.g. ORUs and CMS local bus. This software estimate is based on these adjustments applied to the subfunction estimates provided in the SPDDs. The data in Table 3 is organized according to the SwRI projection of the combined CSCI architecture presented in Figure 9.

While the total source lines of code projected in Table 3 represents a large reduction from other documented projections, the subcontractors responsible for previously released estimates are only now reviewing the impact of architecture changes and the use of standard services on CMS SDP memory requirements. This projection, as well as several recent unofficial estimates, indicates that the downward trend in the DMA memory requirement is being paralleled by a similar downward trend in the CMA memory requirement. As a result, the next official estimate, which takes into account the aforementioned impacts, should indicate that memory usage in the CMS SDP is at an acceptable level, i.e., 3400 kilobytes or less.

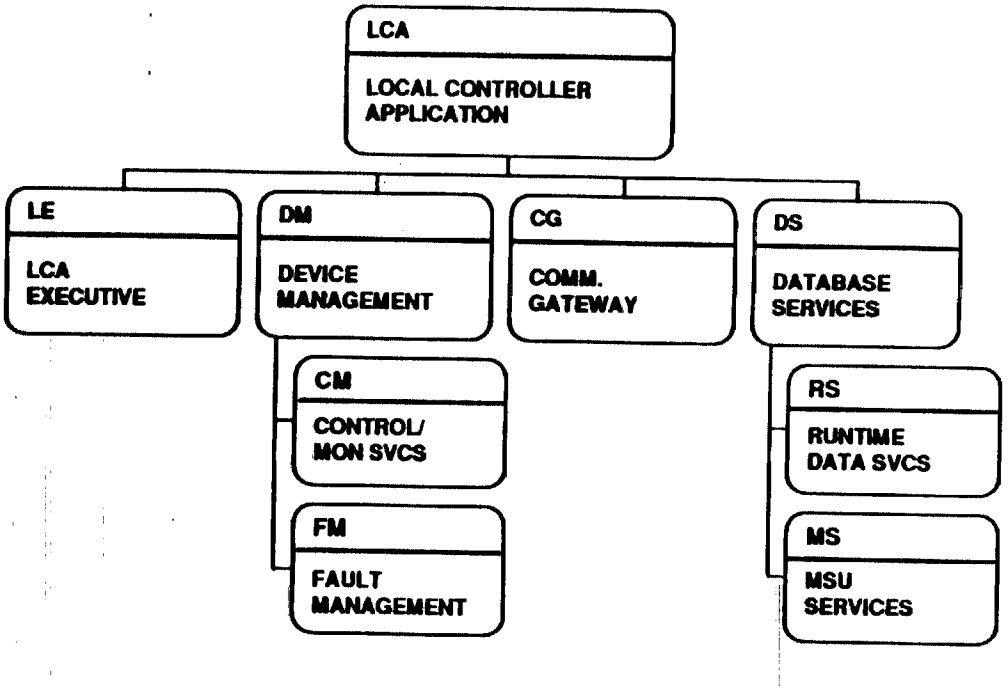


FIGURE 8
LCA SUBFUNCTION DECOMPOSITION FROM THE LCA SPDD

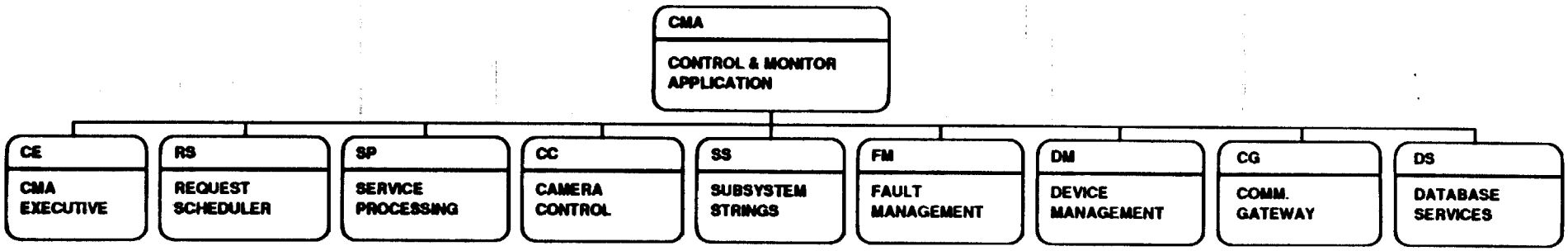


FIGURE 9
COMBINED CSCI SUBFUNCTION DECOMPOSITION – SwRI PROJECTION

TABLE 3
SwRI CMS APPLICATION SOFTWARE ESTIMATE

CMA EXECUTIVE:	715 SLOC, 7.7 KB DATA (10% INCREASE OVER CPA EXECUTIVE ESTIMATE)
REQUEST SCHEDULER:	1875 SLOC, 27 KB DATA (25% REDUCTION IN CODE AND DATA FOR STSV USE AND ARCHITECTURE CHANGE)
SERVICE PROCESSING:	1875 SLOC, 10 KB DATA (50% REDUCTION IN CODE FOR STSV USE, 25% REDUCTION IN CODE FOR ARCHITECTURE CHANGE, 50% REDUCTION IN DATA FOR ARCHITECTURE CHANGE)
CAMERA CONTROL:	750 SLOC, 2.5 KB DATA (50% REDUCTION FOR STSV USE)
SUBSYSTEM STRINGS:	3550 SLOC, 25 KB DATA (50% REDUCTION FOR STSV USE, 50% REDUCTION FOR COMMON CODE)
FAULT MANAGEMENT:	1500 SLOC, 21 KB DATA (50% REDUCTION FOR STSV USE)
DEVICE MANAGEMENT:	3646 SLOC, 87.5 KB DATA (50% REDUCTION FOR STSV USE. SPDD LISTED 350 KB CODE AND DATA. SPLIT WAS ESTIMATED AT 50/50).
COMMUNICATIONS GATEWAY:	1875 SLOC, 30 KB DATA (50% REDUCTION FOR STSV USE. 25% REDUCTION FOR ARCHITECTURE CHANGE SPLIT OF 200 KB CODE AND DATA WAS ESTIMATED AT 60/40).
DATABASE SERVICES:	550 SLOC, 47.3 KB DATA 50% REDUCTION OF CPA DATA MANAGEMENT FOR STSV USE PLUS 10% INCREASE TO ACOMMODATE NON-DUPLICATED FUNCTIONS FROM LCA DATABASE SERVICES.
RODB SPACE:	270 KB (20% INCREASE OVER CPA ESTIMATE TO COVER ADDITIONAL ENTRIES CREATED BY ARCHITECTURE CHANGE)
TOTAL:	16,336 SLOC, 528 KB DATA

2.2.1.2 Standard Services Software

Based on the System Engineering and Integration Trade Studies DMS Performance Analysis Summary White Paper, 90IBMX0032, July 31, 1990, the DMS Standard Services Software (STSV) was not coded but was well defined at that time. The STSV duplicates many of the LCA functions described in the various LCA software documents. Apparently, the STSV performs many more functions than originally anticipated by the C&TS subcontractor. Whether the subcontractor is aware of these capabilities and is considering their use in the CMS applications software is not known. Because of the interrelation of the STSV and the applications software, the C&TS subcontractor should be in close communication with DMS software developers to insure good integration with and utilization of the STSV software. No indication of close collaboration is evident in the documentation.

2.2.1.3 Object Oriented Database

JSC C&T personnel initially expressed some concern that the Runtime Object Oriented Database (RODB) planned for the SSF would impose a heavy overhead for both memory usage and throughput. From the available NASA JSC documentation, it was not possible to determine if this concern was warranted. The recent DMS performance report did address the memory requirement and the response time for RODB activities. Table 4 lists the estimates from the DMS report. These are average figures and according to DMS apply to all applications SDPs. The DMS PRAM RODB estimate for the CMS SDP was 176 Kilobytes. The SwRI independent estimate for the combined LCA CPA application is 270 Kilobytes (see Table 3). The SwRI estimate is more than 50 percent higher than the DMS estimate, but may be no more correct than the DMS estimate, based on how little is known about the combined CMS application at this time.

2.2.2 ORU Firmware

The ORU firmware consists of both the controller firmware and firmware for other embedded processors used primarily for signal processing.

2.2.2.1 Controller Firmware

Since the embedded controller had not been selected for each of the subsystem ORUs (see para. 2.1.1.2), it was not surprising to find that the ORU controller firmware descriptions were incomplete. Controller firmware documents ranged from fairly good, typified by the Software Requirements Specification (C&T SGS IF Switch), spec. no. 10033272, March 1990, to very bad, typified by the SGS High Rate Modem (HRM) Preliminary, DPB-008, January 1990.

Both of these documents were preliminary controller software requirements specifications for ORUs of the Space to Ground Subsystem. The IF Switch specification was a 105 page document describing each function and subfunction of the IF Switch software requirements in sufficient detail to begin design descriptions of each firmware

TABLE 4
RODB SIZE ESTIMATES PER USER SDP

DIRECTORIES:

- GLOBAL DIRECTORY (<1 KBYTE OVERALL SIZE AT EACH SDP)
- LOCAL DIRECTORY
  1,530 OBJECTS * 37 BYTES TO LOCATE DATA UNIQUE TO EACH OBJECTS = 56.6 KBYTES

DICTIONARY:

- 200 TYPES OF OBJECTS * 245 BYTES EACH = 49 KBYTES

DATA:

MINIMUM RAM RESIDENT DATA ASSOCIATED WITH EACH OBJECT IS DEPENDENT ON APPLICATION NEEDS AND PROGRAM REQUIREMENTS. (NOTE: A PORTION OF THE RODB DATA WILL BE STORED ON THE MSU TO MINIMIZE REAL MEMORY REQUIREMENTS)

FROM THE DMS PERFORMANCE DEMONSTRATION, SEPTEMBER 6, 1990

CSCI. All sections of the format for a software specification were adequately addressed, and although some of the Message formats were listed as TBD, a table existed for each message type.

The HRM specification was a 46-page document describing in a general way each basic CSCIs for the HRM. No message formats were given, qualification requirements were not addressed, interface requirements were not in an appendix as referenced, and many of the figures and diagrams were hand drawn or obviously hand modified from other documents.

2.2.2.2 ORU Specific Firmware

In most cases, the ORU specific firmware is to be hosted in the embedded controller. Consequently, the firmware descriptions for the ORU specific functions were made part of the firmware descriptions for embedded controllers. There are exceptions to this. For example, the SGBSP has a total of 8 embedded processors, with several CSCIs distributed across multiple processors. In any case, this firmware was no further along or better documented than the embedded controller firmware. Because the distinction between these two types of firmware was not clear, it was confusing when trying to determine what firmware was intended for controllers and what firmware was intended for other ORU embedded processors.

2.2.2.3 Firmware/Software Integration

No specific mention was found in any documentation addressing the differences between software and firmware and how the two will be integrated prior to installation into the C&TS hardware.

The factors of greatest concern for ORU firmware are the lack of procedural descriptions for software integration, testing, and configuration control of the firmware. Typical of the vague planning for integration and test is this statement appearing in several ORU firmware requirements specifications, "Successful completion of software integration and CSC testing are entry criteria for CSCI testing and, therefore, qualification requirements." This statement made no sense to SwRI reviewers, read in or out of context.

2.2.2.4 Firmware/Hardware Integration

Firmware/hardware integration was even less defined than firmware/software integration. Firmware/hardware integration requirements are acknowledged in documents such as the Space Station Software Standards Document, JSC 30244, September 1986, but are not adequately addressed. For example, JSC 30244 states, "The Firmware Support Manual (FSM) documents are the primary means of determining how to develop, support, and maintain the firmware." But, the accompanying FSM outline does not even mention integration with hardware or software.

The Software Verification & Validation Plan (G.E.) Appendix 1, MDC H4414, May 1990, in paragraph 4.4.2 states that the firmware will be card level integrated and that

test will be used to verify test requirements, but no specifics are given. In paragraph 4.5 of the same document it states that C&TS subcontractors are not required to install or check their firmware. The C&TS contractor is supposed to install, integrate, and check out the complete system once the firmware is received from the subcontractors. Yet, no details for this activity were evident in the documentation.

2.2.3 Software Language

2.2.3.1 Storage Space Considerations

One of the driving software considerations in this assessment was to meet an objective of estimating the amount of actual memory required by a collection of unwritten Ada programs for the Communication and Tracking Subsystem (C&TS) on the Space Station Freedom (SSF). In approaching this problem, three complicating factors occur. These factors are separate from the difficult problem of actually estimating the amount of Ada code necessary to complete the application programs. Once the Ada code estimation task is completed, then the following three factors will bound the memory space needed for that Ada code. The factors are as follows:

1) The hardware employed for execution.
2) The compiler used for the application.
3) The software development procedures followed during implementation.

It is not possible to make an accurate estimate of the amount of memory that will be used for a function without establishing values for these factors. Hence, the estimation of required memory storage must include critical assumptions when these elements are not known.

In the case of SSF, the hardware device is known, and it will be either the Intel 80186 or 80386 processor, depending on the SSF unit being considered. The Intel data sheet on the 80386 processor states that the average instruction length is 3.2 bytes. Hence, a reasonable value for the hardware conversion factor can be gleaned from the manufacturer's data. Since the choice of processor is no longer under consideration, it will not be examined further.

The other two factors, the choice of compiler and the software development procedures, are unknown at this time, and they have a very strong bearing on the final storage requirement estimate. The difficulty of the space optimization, which depends on the memory space use estimation, can be understood by noting that even with the best accuracy of the assumptions available, the estimated memory space required is gauged to range at least three to one/third times the actual code memory storage space required during operation. This means that the estimate could easily lead to developing code that could not possibly function. In this situation, it is not sensible to simply assume that the memory storage can be expanded to meet the need. It is understood that the processor on the SSF has the basic capacity for increased RAM memory, however there are a large number of real-world constraints that must be met before the expansion can become a reality.

This section discusses the compiler and software development method factors; however, another consideration must be addressed first. This parameter is execution time. An important consideration is that memory storage must normally be balanced against the execution time. That is, if the memory space is the key optimization factor, then the execution time of the software will most likely be increased from optimal. It also follows that when the execution time factor is being optimized, memory requirements are often increased from optical. This is not an absolute nor linear rule in every case, and when both factors are equally significant, it implies that reaching an optimum balance results in approaching minimum execution times, while also attempting to approach near minimum memory storage space. This is obviously a difficult, but worthy engineering goal.

It turns out that performance is normally thought of from the execution time point of view, and when most systems are said to be optimized, they in fact are approaching minimum execution times. Storage space optimization is often left to be done by hand coding procedures, or possibly requested by a single option on the compiler. In the latter case, it is always accomplished at the sacrifice of execution speed. Because the execution time is the better known and the more standard optimization target, the optimization of storage space side of the performance equation is all that will be addressed. This section was written to provide that spotlight for the space factor to insure appropriate questions are asked so that the correct answers can be obtained, and correct direction can be given throughout the software development. This is much different than providing a systematic method for analytically quantizing these values into an estimation of the space use. However, specific benchmarks to be used to help estimate storage space are recommended. If the questions raised in this section are presented to the developers, and the answers are unknown or cannot be adequately addressed, then the worst case storage situation can be expected.

2.2.3.2 Compilers

The performance of a compiler is a function of a multitude of interrelated performance considerations. Experts in this field have stated that no single approach to evaluation addresses the requirements of everyone who needs to measure performance (Dongarra, Martin, Worlton, 1987). Yet some gauge of compiler performance is necessary to make tradeoff decisions.

Studying the results of several benchmark and Ada evaluation projects, it becomes very clear that there is a large variability in the amount of storage and the execution time required for an application program depending on the particular compiler employed. The standard method for comparing the storage performance of two different compilers is to complete and execute a selected set of application programs and compare the storage space needed. The ideal case would be to develop the operational program being considered and simply compare total storage space once compiled. This is obviously an impractical situation, in that the development effort would be complete before an estimate could be made, and this would still not consider storage space performance of future programs. A more reasonable choice would be to obtain a set of universal programs that can execute on the machine being used and then compare the storage space resulting from different compilers to get a relative storage performance indication.

Programs used in this manner are referred to as a benchmark grouping, and simple analysis of published benchmark storage requirements for the target machine is not an unreasonable approach at obtaining a first cut estimate of the relative storage space requirements for different compiler systems. Unfortunately, the state of benchmarking is confusing and can often provide inconsistent projections. Benchmarking difficulties arise as the overall performance is improved through optimized hardware system organizations. The advancement of the hardware technology causes most complex architectures to do extremely well on one kind of a benchmark problem, while doing poorly on another seemingly equally valid benchmark program. To be an effective estimate, the nature of the benchmark programs should match the nature of the application programs.

The adoption of a particular compiler is loaded with problems, and before the undertaking is completed, it is recommended that the Carnegie-Mellon University Software Engineering Institute's "Ada Adoption Handbook: Compiler Evaluation and Selection" Technical Report be reviewed for excellent guidance in the choice. In fact, SEI has a synthetic benchmark called Hartstone that is targeted for hard real-time applications. The benchmark source code and other information can be readily obtained from SEI at no cost.

There are several ways that the compiler choice influences the storage space. The particular compiler/linker combination drives the final optimization possible. Some of the ways storage space is affected are the efficiency with which the assembly code is developed, the methods used for storage reclamation, the handling of complex data types developed by the user, the methods of developing the dynamic storage called the heap, the allotments for the stack, and many others. As an example of space variability, the benchmark provided by the Software Engineering Institute at Carnegie Mellon University discussed above has approximately 1,000 lines of Ada code. As compiled and linked data it took from a low storage capacity of 30,000 Bytes of memory for DEC VAX Ada 2.1, to a maximum storage capacity of 267,000 Bytes of memory for the DDC-I DACS-80386PM 4.3(3.1a). There were seven other compilers in the evaluation and there was a fairly even spread between these extremes.

As it was pointed out, optimal performance usually refers to minimum execution time, therefore most benchmark data available is designed to provide detailed information on the optimal "execution time" and only indirectly and briefly presents the storage space information. Storage space requirement is usually one paragraph placed somewhere near the end of the report that will not have been developed with as much information as one normally desires. Since a space discussion is hard to find, and usually not complete enough to be understood, the following information is designed to clarify some of the more significant storage space considerations.

Depending on the Ada compiler chosen, several factors characterize the compiler effectiveness which influence the space use. The final performance space utilization capacities can vary greatly between compilers based on the objectives designed into the compiler.

Estimating storage space becomes quite complicated because user defined program types are well supported and well used in Ada software. Besides the normal simple types such as INTEGER, and FLOAT composite types may be made up of multiple values. In fact, composite types may be made up of values that are themselves composite types. Further, a legal object in a program can be a task, and tasks can be considered composite objects which are executable. All of these factors are complicated by the fact that these can occur on entry to a subprogram where the object dimensions are not known until run-time and or in the declare block that is statically predetermined. The storage space provided to the program as storage area for dynamically sized and allocated objects is called the heap.

Dynamic allocation can be accomplished to cause optimization. Declare blocks are commonly used when some of the objects are of undeterminable dimensions. In this case, the rule is that the maximum possible size is allocated which is clearly a space over time use tradeoff.

The "pragma" PACK and the "attributes" STORAGE_SIZE and SIZE also impact storage allocation strategies. PACK tells the compiler to use space minimization as the criterion when selecting a format for representation of a variable or array of variables. This may also affect the overhead involved in manipulating objects of this type which affects space and execution time.

STORAGE_SIZE and SIZE, in counter balance to the pragma PACK, can be used with representation clauses to command the compiler to use designated sizes for object representation and storage availability. Changing sizes using these attributes will negatively affect run-time overhead whenever allocating or manipulating one of these objects.

Three specific compiler domains that directly influence the storage requirements which are associated with chosen compiler strategies in the implementation are the stack, the heap, and the code.

2.2.3.2.1 The Stack

The stack is a scheduled area for storage where last-in first-out (LIFO) allocation is employed dynamically according to program structure and subprogram calling paradigms. The parameters of the stack that should be quantified are:

1) Size - limits the level of recursion possible, and sets the thresholds to convert stacks into heaps.

2) Usage - Items that are normally allocated from the stack, may instead be allocated from the heap or the base of the stack at compile time. Data is normally stored on the stack. This is necessary for data contained in procedures which are potentially recursive or so that each activation will have its own copy of the data. On some machines, accesses to the stack are very efficient and there is no penalty involved in referencing data that is on the stack. On other machines, direct addressing of memory is faster

and more compact. Hence optimum compiler choice related to the stack depends on the hardware system.

Another stack consideration is the compilers method of implementing the "cactus stack". Ada is a multi-tasking system, so the total stack allocation is managed as a collection of substacks, one for each task. This organization is called the cactus stack. The cactus stack organization is important because the way that the several task substacks are linked together will impact the performance of the non-local memory references.

2.2.3.2.2 The Heap

The heap is a storage basin accessible for dynamic program application when the space needed is not suitable for the stack because it is too large or is used in a way that does not accommodate the LIFO discipline used in the stack. The way the compiler uses the areas of the heap that should be quantified to compare compiler capabilities are:

1) Structure and Management - Working size of the heap, and the compiler's policy for increasing the working size of the heap. The interaction of both the physical and virtual memory space with the heap determines the overhead associated with increasing the heap size.

2) Size - Support for deallocation, either explicit by a call to a subprogram that relies on the run-time system to return the space to the heap, or implicit by the use of a garbage collector that is a process running concurrently with the program. The best choice is dependent on the application program.

2.2.3.2.3 The Code

The memory space needed for the code is directly determined by the compiler's methods and efficiency of converting to the machine language used by the hardware. The area of the code that should be quantified is code generation efficiency. The effective use of registers, instruction reordering, and other compiler optimization factors can reduce code size greatly. Sometimes, while providing for faster execution time, code size may be increased by reserving space within the code segment. Since embedded systems are usually known to emphasize efficiency in time and control at the cost of space, reduction efforts in space, usually require more indirect calls, and memory references, which lower performance. This is essentially, the space and performance antithetic goals as were discussed earlier. In a tightly designed system, a careful software engineering analysis needs to be completed to optimize a software system to meet the constraints of the system storage space and simultaneously meet the execution time goals. However, the incorrect choice of compiler and its lack of optional features related to optimization could disallow this engineering option.

Compile time for an embedded system compiler may be greater than that for a development environment, but the execution time for the embedded system may be much less. Development of Ada code can be hurried by choosing a compiler that speeds through the compiling activity in preference to requiring multiple sweeps through the code to allow optimization of the machine code memory utilization. Choosing compile time over space utilization, would be an incorrect choice when the system is expected to be constrained in the spacial arena.

Also, it should be noted that there is a basic difference between program-level storage, and run-time system storage. Program-level storage elements are the constants, variables, and execution code declared by the programmer, where run-time system storage involves the dynamic data structures used to support the execution of the program and uses a variable amount of code space. When determining the memory space required, it is important to characterize the size of the maximum run-time space utilization scenarios.

Finally, a specific recommendation on C&TS can be suggested. From the information obtained on NASA's evaluation of a large number of benchmark programs, it appears that the compiler/linker for SSF is being chosen with minimum execution time goals as the performance factor. If this is the case, it would be prudent to include the software generation methodology a guide to code for minimum space utilization. The overall discussion of the software generation methodology is the next topic.

2.2.3.3 Software Generation Methodology

During the process of software development, many moments will exit where the programmers choice will greatly influence both the execution time and space utilization. The extent that this process is optimized is directly related to the extent that the software is developed under correct software engineering procedures. There have been many successful projects that have met all of its requirements without rigorous or even crude application of a software engineering methodology. These projects are often successful only because the inefficiencies caused by improper coding procedures are camouflaged by the expansion of memory, acceleration of the hardware components, or simply because of a very permissive set of requirements. However, only a small percentage of software systems developed have had the same level of space and execution time performance requirements as have been imposed on the processing activities that support the C&TS of the SSF. Hence, the conclusion is that the NASA contractors should follow a very rigorous approach during the development of each and every piece of software.

It should be noted once again that the software design will need to minimize space and maximize processing speed at each step of the process. In fact two separate architectures could be developed at multiple points, with clear space/speed tradeoffs evaluated at these times. The final design would most probably need to take advantage of leverage situations where large gains in one direction, say less memory storage requirements, yields only a small loss in the other direction, say slower execution. The key to the design process is the knowledge of the existence of a trade-off and a quantified measure of the improvements made to assess each situation. A great deal of prediction

and benchmark measurement need to be made as the solution is coded into its final Ada form. Each space improvement needs to be weighed against the speed cost. Only in this way will the system be simultaneously small enough to fit in the memory allocated, and still have the execution time performance to meet the established requirements.

Overcoming this rigorous software generation problem is no easy task. NASA and its contractors have established the required standards and directives. The following is only a partial listing.

1) MIL-STD 2167
2) NASA SSSSD 30244
3) Software Product Assurance Plan - MDC-H4040
4) Software Management Plan - MDC H4058
5) Software Development Plan (Houston) - MDC-H4077

It is clear that there is no shortage of software methodologies, the only concern is in their application. The key is to be assured that the software development plans are being followed, and they are not simply a paper generation activity, that finally stops once the code development begins. The way that this can be accomplished most effectively involves two basic steps.

The first step is to use a method to determine the contractor's capacity to conform to the software engineering development plans. It is not reasonable to simply assume this is true, because most contractors who have been evaluated in this area fall short having the required level of capacity. The Software Engineering Institute at Carnegie Mellon University has developed a checklist method that can be used to compare software engineering methods and procedures against an optimal set of software engineering guidelines. This document contains a lengthy set of questions and is an excellent guide to assess the engineering software development capability of an entity. Normally, this checklist would be used to prequalify a contractor who wishes to develop software for the government. In this case, with the contracts and subcontracts already awarded, it is still reasonable to request the assessment of the contractors to determine their software engineering capabilities and to decide upon further actions to guide the development to success.

The next step to successful software development is a series of repeated actions to assure the software method is actually being used during software development. This requires cross reference data such as verifying stability of the software managers of the project, communications from the software quality assurance function, evaluation of the software training programs first hand, attendance to formal management periodic reviews, and the list goes on. The point is to ask the contractor specific questions, review specific documents, and verify specific activities to independently measure what is happening during the development of the software.

The plan to complete this activity is the Verification & Validation Plan, MDC-H4249. The Software Integration and Test Organization (SITO) of the McDonnell Douglas

Space Systems Company, Space Station Division-Houston is responsible for carrying out the plan as an "independent" reviewer. It is not always within the government's best interest to have the same company do the IV&V that develops the software.

Since the SITO conducts formal tests and writes test reports which include test requirements, test management, test methods, test data requirements and acceptance criteria, review of these reports would be invaluable to determine the effectiveness of SITO. A list of some of the reports that should be obtained and reviewed includes:

1) SITO Verification Test and Analysis Reports.
2) SITO Review Item Discrepancy (RIDs).
3) SITO Discrepancy Reports (DRs).
4) SITO Reports on testing showing all anomalies.
5) SITO's collection of the Test Description Sheets (TDS).
6) SITO's review of deliverable documents.
7) SITO's developed checklists of questions for specific Unit test
8) SITO's developed checklist of questions for specific CSCs.
9) SITO's developed checklist of questions for specific CSCIs.
10) SITO's developed checklist of questions for system acceptance.

Review of the SITO documentation, especially the checklists, should begin very early in the program, because a weak IV&V program will result in a poor final product. The cost to improve the final product, will be orders of magnitude greater than the cost to provide proper IV&V.

2.2.3.4 Software Integration

A software integration plan is included in the Software Development procedures. The specific implementation of integration plans for the SSF is described by Software Engineering and Integration Plan, MDC H4421.

This part of the assessment is not intended to point out weaknesses or strengths in that plan. The most important question that can be asked about the plan at this point, is the same that was asked about the software development procedures, that being, is the plan being used.

There is a tendency to believe that integration occurs as the final step of a development. In fact, if the software integration does not begin before the code is developed, it has begun too late.

SSF has multiple contract services, with multiple agreements. In order to be effective, integration requires a responsive, independent single point integrator, and a completed test plan. The independence at this level refers to being independent of the code developers, not of the company, as is recommended for IV&V. However, this is not the plan for the integration of the CMS CSCIs, CPA and LCA. There is involvement of other organizations such as the Software Verification and Validation (SV&V) group, the

Systems Engineering organization, and the Software Product Assurance (SPA) group. The outlined responsibilities state that, after the last build has been validated, software is turned over to the System Engineering organization for C&TS integration testing and, in conjunction with NASA acceptance testing.

The key documents that would be of interest to NASA are associated with the integration test plan. The test plan must be formal, well developed, and available. The Software Development Files (SDFs) are required to have the test requirements, test cases, and test procedures for the test entity(ies), hence they should be available long before the test reports. Reviewing the integration test plan would be a method of evaluation of the integration process, before it actually begins. This allows a correction phase to occur if it is determined that the integration process is not on the right track.

At this point, it can be stated that there are several subjective circumstances that indicate integration may be a problem area. Some of these circumstances are:

1) Significant overhead time required for code revisions.
2) Test plans with TDB detailed specification.
3) Lack of evidence of formal communication channels.
4) Significant errors in documentation and report.

2.2.3.5 Source Lines of Code (SLOC)/Byte and External Factors

The last topic to be covered under software language is associated with the often quoted SLOC/byte ratio. In an effort to determine storage requirements for the Ada code, it has been determined that it is possible to estimate the lines of Ada code necessary for a function. Having a ratio to convert the lines of code to bytes of memory would then allow an estimate of required memory. Throughout this assessment, several values have been discovered, and used by the contractor. These ranged from a reported low of 16 to a high of 30. Since the compiler has not been chosen, it is obvious from the compiler data in the section above, that the range is much greater than 16-30, and will probably be much higher in the final analysis.

The difficulty in the search for a single ratio can be seen in the fact that there are several external factors that also influence the ratio. These factors are:

1) DEFINITION OF A SLOC. This varies from the number of carriage returns to the number of active semi-colons. Usually comments are not counted, but vague references to the number of Ada statements can be off by a factor of 2 or more from another definition.

2) RUNTIME OR STATIC STORAGE. Runtime storage is much harder to determine than program level storage. Obviously, the maximum runtime and static storage is the requirement, but usually the static value is the only one given in a study. The swing between these two can be very small or very large depending on the program.

3) OPERATING SYSTEM. The choice of operating system effects the storage size, because the Ada code must interact with the operating system. The differences here are usually smaller, but measurable.

4) NATURE OF CODE. If the code has a lot of data storage, a lot of I/O, or computational requirements, then the ratio of the Ada code to machine language will be different for each case. The mix of each of these types in the application code has a large affect on the ratio.

5) METHOD OF MEASURING SPACE. There are no established procedures, just conventions used in circumstances, after assumptions are made. Using the difference between the ADDRESS attribute of two labels is often used, but not all implementations support the ADDRESS attribute. A different ratio would result depending on the method used.

Combined, these external factors can be as significant to the swing in storage space required as the choice in compiler or the way the code was developed. Essentially the conclusion is that one should be very wary in applying a single ratio as the method of predicting how much memory space will be required for a given situation.

3.0 SYSTEM ASSESSMENT

Once the available hardware and software documentation was collected and understood, the assessment process was begun. The first step in the actual assessment of the C&TS was system definition, then development of an assessment methodology, followed by formulation of assessment considerations.

3.1 System Definition

A system definition is necessary for establishment of a baseline from which to start the assessment process. The baseline consists of the requirements by which the assessment must judge the adequacy of the system and the system design which is to fulfill the requirements. Unfortunately, the SSF C&TS has changing requirements with a design lagging in response to those requirements.

3.1.1 Baseline Configuration

SwRI personnel spent several weeks trying to establish firm requirements and a corresponding C&TS design from the contractor documentation and JSC interviews. Eventually, attempts to formulate a firm baseline configuration were abandoned when it was realized that many aspects of the C&TS were constantly changing. This was a significant decision with several negative consequences.

3.1.2 Consequences of a Variable Baseline

Since the requirements and design could not be defined, except in general terms, most quantitative results would have minimal value. Due to system complexity and the close interaction between subsystems, even the effect on numerical values could not be estimated with any certainty.

Because the documentation and other information lagged the C&TS change process, the assessment process would always be behind. It was estimated that even the simplest agreed-upon change required a minimum of two months before the corresponding directive reached all concerned C&TS organizations. Time for implementing the change was never considered. Since it was observed that significant changes were occurring at a rate faster than one change per two months it is possible the design never reflected actual system plans at any point in time.

If true, any assessment based on changing requirements and a nonresponsive system design would have minimal usefulness in meeting the traditional goals of design verification and optimization. On the other hand, such an assessment may be very useful for drawing attention to potential problems and overlooked issues.

3.2 Assessment Methodology

To perform the assessment it was necessary to develop an assessment methodology. This was also a primary goal of the C&TS assessment task. Any methodology developed for the C&TS of the SSF was to be useful for other manned spacecraft. The initial project plan described in Section 1.2 was the starting point for the evolution of the procedures that eventually were used for the C&TS software assessment. The initial plan basically described two features; data collection and a "two prong" approach to hardware and software.

3.2.1 Data Collection

Since the SSF C&TS was an unknown to SwRI, data collection was the necessary starting point. This would probably be the starting point for any similar assessment performed by an outside organization. Once the data sources were identified, the data was collected and reviewed for currentness and coherency. Only those data sources which were both current and coherent were to be considered during an assessment. This was difficult to do for the C&TS. Most documents dated prior to the C&TS PDR (May 31, 1990) were of questionable value, but in some cases they were the only data sources available. At one point, a cut-off date of January 1990 was used in an attempt to limit the assessment dependence on very outdated information. This did not prove successful since many of the more recent document referenced older documents that were incomplete or outdated.

JSC personnel had asked that documentation holes and discrepancies be flagged during the data review. This seemed possible early in the assessment, but as the assessment progressed, it was apparent that the reviewers could not determine what was correct, current, or accurate. This was the result of many factors, but the most serious was the lack of good document control at JSC. There was no structured way at the JSC to determine the currency of documents apparent to the SwRI assessment team.

3.2.2 Two Prong Approach

For the C&TS, it seemed that the software allocation should be addressed by investigating the requirements and design for software separate from the requirements and design for hardware and then to resolve the discrepancies between them. This was an arbitrary approach which appeared to be as useful for performance of the assessment as any other approach. Of course, the changing requirements and lagging design changes of the C&TS may have masked the effects of one assessment method over another.

A series of assessment flow charts were developed in an attempt to provide a methodology that might be useful on future programs. The charts are shown in Figures 10 through 12. Figure 10 shows how a software allocation assessment might be performed if the assessment was begun with the design process and system requirements were fixed for the duration of the program. Figure 11 shows how a similar assessment might be performed on a program where design was already begun, still with fixed design requirements. Figure 12 is representative of the C&TS software allocation assessment, in

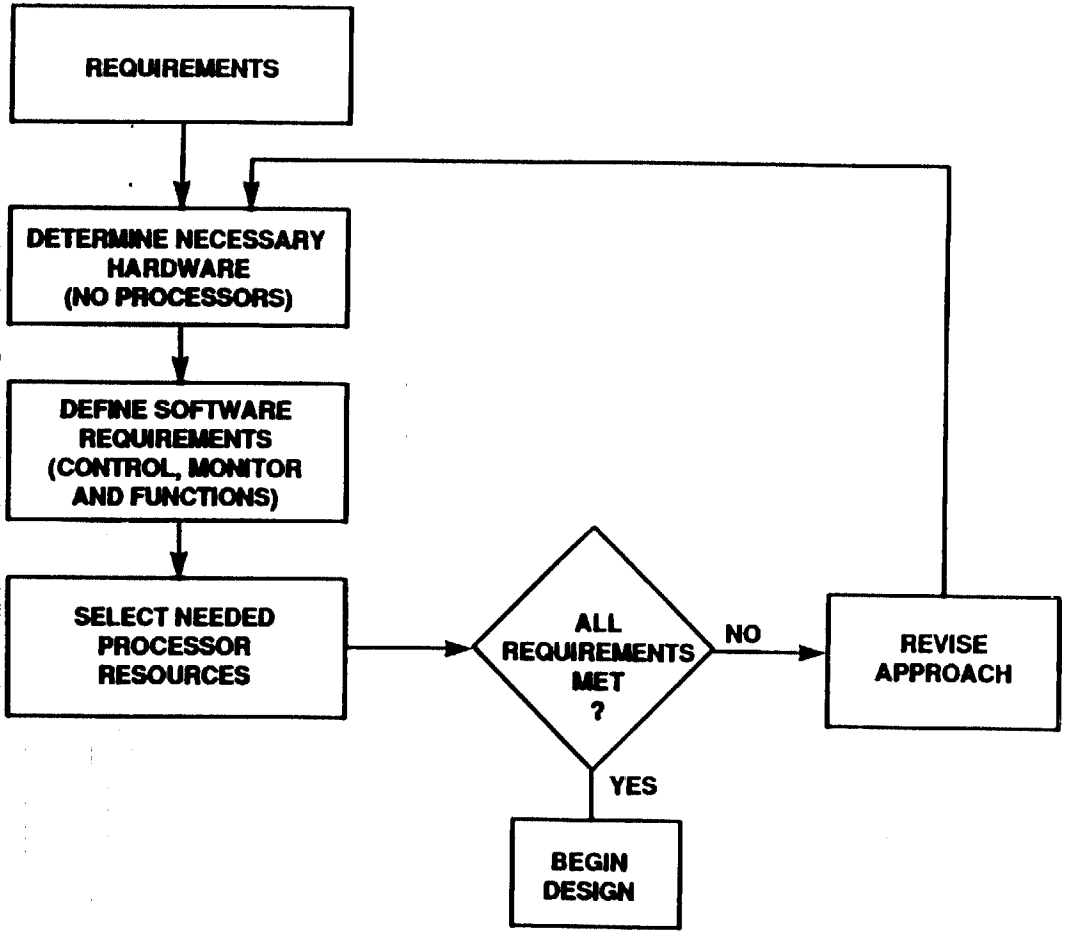


FIGURE 10
ASSESSMENT AS PART OF THE DESIGN PROCESS FOR
DEVELOPMENT WITH FIXED REQUIREMENTS

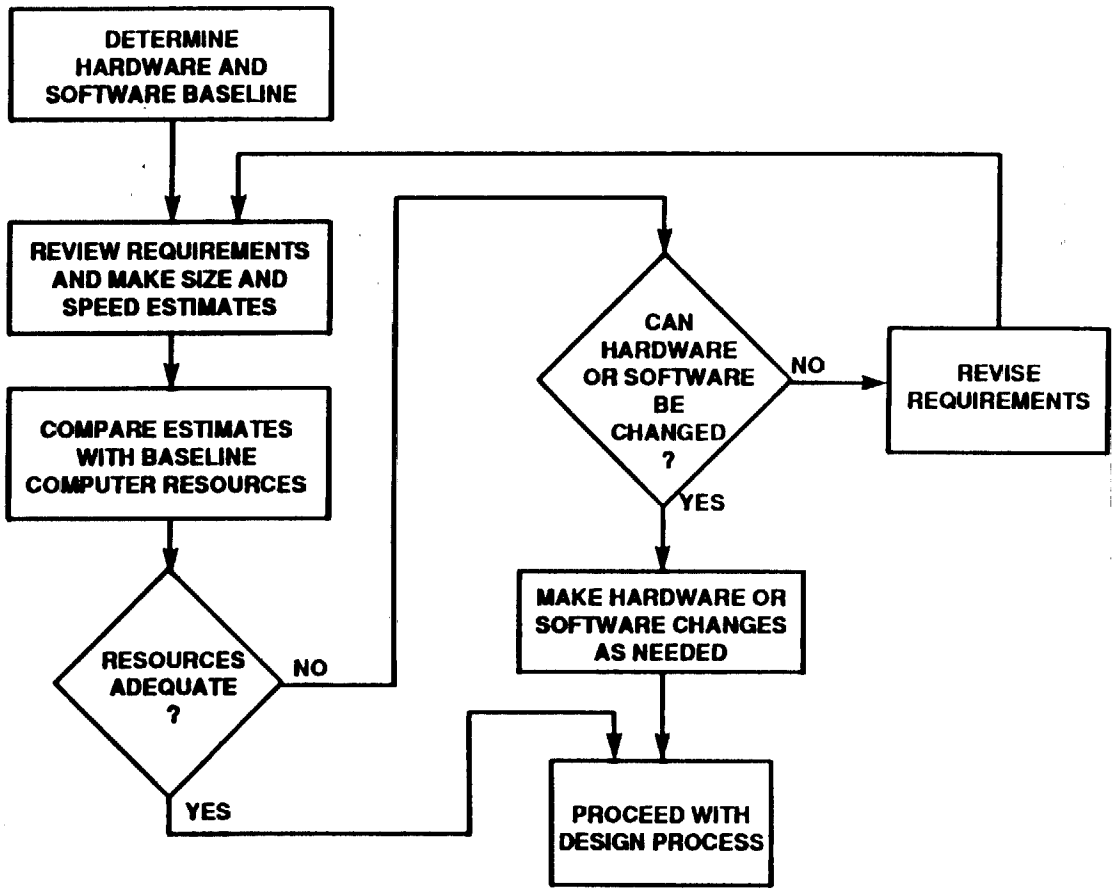


FIGURE 11
ASSESSMENT AFTER DESIGN BEGUN FOR DEVELOPMENT
WITH FIXED REQUIREMENTS

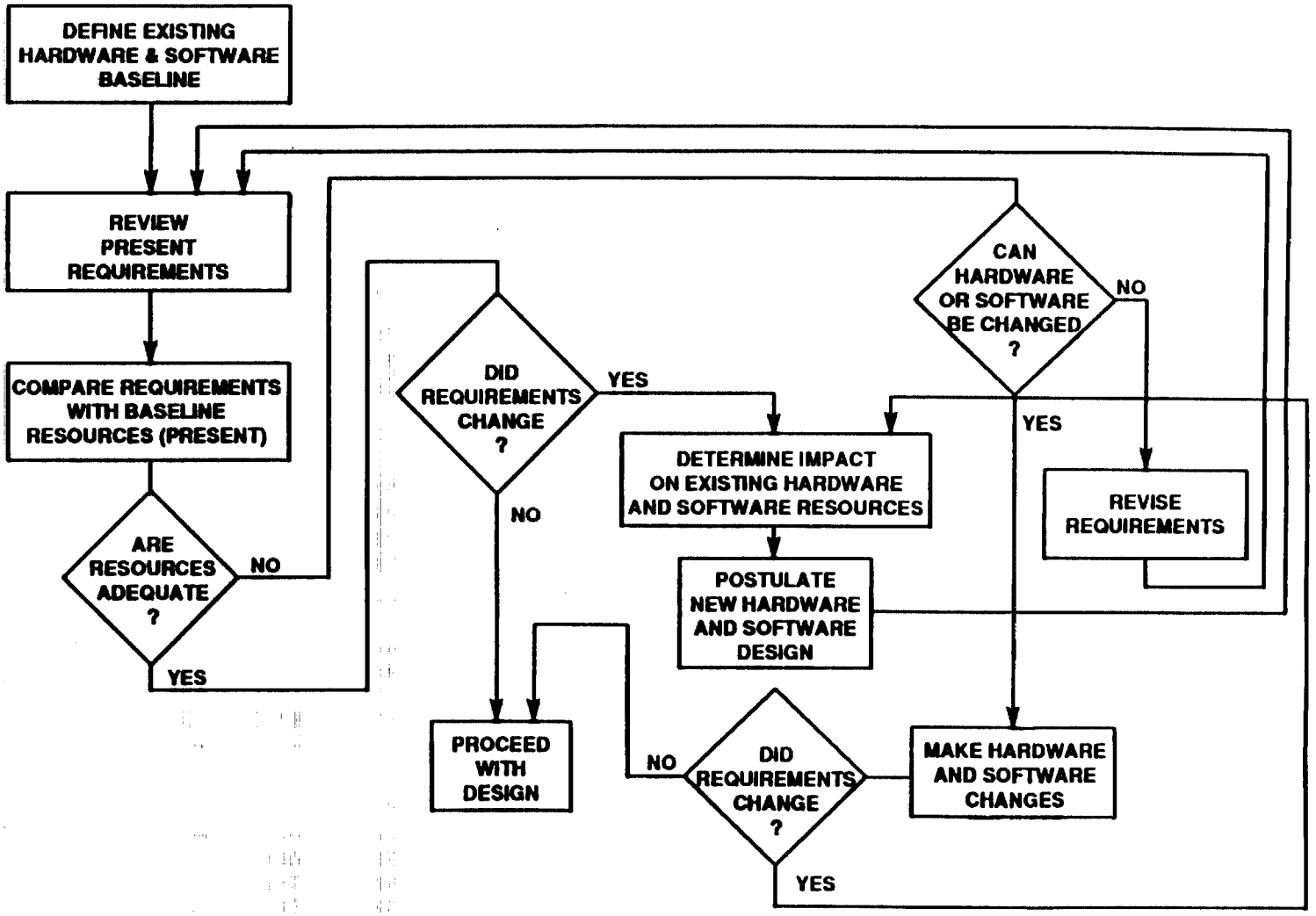


FIGURE 12
ASSESSMENT FOR A DESIGN IN PROGRESS
WITH CHANGING REQUIREMENTS

that it attempts to show how an assessment might be performed on a system with design started while design requirements are changing. Because of the many nested loops it is obvious that the assessment process, as shown, could never proceed with design until the requirements remained unchanged.

3.3 Assessment Considerations

Once an assessment methodology was developed it was necessary to determine the assessment factors that must be resolved to provide a practical software allocation. Since the C&T System's primary limited memory space for the necessary software, and data throughput, the factors influencing these constraints were the main consideration of the assessment.

3.3.1 Factors Influencing Memory Requirements

The factor most often considered in the C&TS software documentation was the size, in bytes, of the code resident in the various ORU processors necessary to perform the C&T functions. This seemed appropriate if code size could be determined to any level of accuracy prior to writing the code. Most of the processor hardware memory capacities were defined early in the program, so it was basically a question of whether the needed software would fit within the planned memory.

Most attempts to determine the memory requirements for the code started by estimating the source lines of code (SLOC). This estimated value was then multiplied by a conversion number to give the memory size necessary to hold the compiled source code, ancillary files, and other overhead. It was difficult estimating source code size for any processor since the requirements, and implementing hardware were changing. It was even more difficult to determine a realistic conversion factor when the compiler, source language, and software methodology were all in question. As pointed out in section 2.2.3, the compiler and software practices are as important in determining memory requirements as the amount of source code.

Unfortunately, the conversion factors, could easily vary by an order of magnitude and could not be determined to any better accuracy than that at this stage in the software development process. This lack of a reasonably accurate method of determining code size meant that all memory requirement estimates, done prior to coding, could easily be very wrong.

3.3.2 Factors Influencing Throughput

Throughput estimates were more nebulous than the memory requirement estimates. As pointed out in 2.1.2.2 the C&TS contractors had made several estimates of local bus throughput, signal throughput, control response, and other time related performance parameters. In most cases reviewed, the conclusions were presented without backup material. And in the few instances where the material was presented, it appeared incorrect or did not fit the present design scenario. The conclusions the contractors reached

regarding throughput may or may not be correct. Insufficient information was available to determine the validity of their estimates.

System throughput is influenced by all of the factors influencing compiled code size as well as other factors such as processor speed, code complexity, and processing "bottlenecks." All of these variables were so poorly defined at the time the assessment was in progress, that no definitive statements could be made about the system throughput, except that some potential problems were apparent to the SwRI assessment team.

3.3.3 Other Factors

If all the factors mentioned in 3.3.1 and 3.3.2 were not enough, there are several factors that may influence both the memory requirements and the system throughput. For example, control of the software development process in areas other than coding techniques could be important. Considerable improvement in memory requirements and throughput might be possible from intersubsystem synergism resulting from careful control of software integration between systems and subsystems. Other possible factors of importance are the RODB overhead, impact of Commercial Off-The Shelf (COTS) software on the complete system, and code structure (common software, priority handling, etc.).

4.0 CONCLUSIONS

Although the initial assessment goal of allocating the correct software to the available processor resources was not met, several conclusions important to the C&TS development program were reached and an assessment methodology for similar programs was developed.

4.1 Assessment Value Limited by Changing Baseline

It became clear as the C&TS software assessment progressed, that the assessment value was hampered by the lack of fixed system requirements and a changing design. C&TS documentation was either incomplete or inaccurate because the contractors were continuously revising the design in response to changing requirements. Despite the contractors' efforts, they were always behind the change process for the duration of the software allocation assessment program.

4.2 Memory Requirements

The memory requirements were being influenced by the changing system requirements. Since most design changes ("scrubs") resulted in reduced hardware or operational requirements, the amount of software necessary to meet the requirements was reduced. With each downsize of the system, the local bus throughput was of less concern and the number of controllers to be addressed by the CMS was reduced. Even the elimination of the second SDP from the C&TS design, which resulted in a net reduction in available memory, was balanced by a downsizing effect on the CMS software.

There was some concern that the software originally planned for the C&TS SDP may not fit within available memory (4 Megabytes). If it can be realistically utilized, there is more than enough expansion capability (to 32 Megabytes) to cover any software currently planned. This seems to be a common situation for all the embedded processors/controllers. With the exception of the C&TS SDP, there seems to be more than enough basic memory hardware within each embedded processor/controller to cover anticipated software size. But, as a precaution, expansion capability has been planned in most cases. Because of the expansion capability and the way the processors have been integrated into the system design, they are not easily limited in memory capacity. The remaining concerns would be the effect on throughput and the impact on the hardware design due to increased memory (power, weight, etc.).

It is doubtful that any memory increase will be necessary if improvements in coding practices and compiler efficiency are implemented. Nothing in any of the documents indicated that there has been an attempt made to optimize the coding or compiling efficiency. So, concern for exceeding available memory space is prudent, but probably premature at this time.

4.3 Throughput

As mentioned in 2.1.2.2 very little can be said about the true signal or command throughput for any part of the C&TS until the software development is further along. But,

there are a number of potential problems that should be carefully watched during the software development process. These include prioritization of tasks, timing between concurrent tasks, and local bus limitations.

It was noted in some of the compiler data from sources outside the C&TS design program that the priority assigned to various activities within real time software does not always result in the code execution sequence planned by the programmer. Unless careful tests of all prioritization schemes are performed after each source code compilation, there is the possibility that the prioritized tasks will not be performed within planned limits.

If flaws occur in the priority structures, this could cause problems with the timing requirements between concurrent tasks performed on the same or independent processors. Timing is also a critical factor for data and control messages on the local bus. There must be complete understanding between the various users of the bus so that proper interpretation of all timing parameters and requirements is made. This is a critical integration factor that can only be adequately handled by careful control of integration issues during the entire software development process.

4.4 System Flexibility

Despite all the pitfalls, the basic design of the C&TS appears to have enough inherent flexibility to expand or contract in any direction. Without this flexibility, the probability of failing to meet the finally established requirements would be high. Although the system design may be flexible enough to cover contingencies, if this flexibility is used, completion of the system will cost more and take longer than planned. The flexibility will be best used if there are careful integration control and good channels of information exchange between requirements managers, designers, and design implementers.

5.0 RECOMMENDATIONS

Some recommendations have resulted from the assessment process that may be of value to the C&TS design effort. These recommendations are briefly described in the following paragraphs.

5.1 Common Controller Software

One common feature of the ORU embedded controllers was that all controllers performed several basic functions in addition to unique ORU functions. These basic functions are presently being addressed independently by three C&TS subcontractors. It would seem that a common software module could be developed to handle these similar functions for all embedded controllers. The ORU specific applications software could then be added to this common CSCI prior to compilation for the firmware. This has two advantages. The software has to be developed only once and any necessary debugging fixes all controllers. It has an additional advantage in that it gives the subcontractors an opportunity to communicate on a topic of mutual benefit.

5.2 Reduce Controller Hardware

Another observation was that the need for an embedded controller in each ORU is not clear. No reference was found in the documentation explaining why a particular ORU required an embedded controller. Also, most embedded controllers seemed to have some underutilized features (mostly excessive ROM).

With the exception of the video subsystem, most ORUs of the C&TS have embedded controllers. In some cases, such as the antenna controller for the Space to Ground Subsystem, this limits the ability of the antenna controller to influence the operation of the receiver/transmitter, making antenna control more complicated than necessary. If the controller in the Space to Ground Transmitter-Receiver (SGT-R) were replaced by hardwired control from the Space to Ground Antenna Controller (SGAC) this might simplify Space to Ground Subsystem (SGS) operation and reduce hardware (less weight and power). This same scheme may be possible with other C&T ORUs such as the Assembly/Contingency Baseband Signal Processor (ACBSP) and the Assembly/ Contingency Transmit/Receive Amplifier (ACTRA), or between the Space to Ground High Rate Multiplexer (SG HRM) and the Space to Ground IF Switch (SGIFS). A trade study should be performed for all C&T ORUs to determine if hardwired control from a collocated ORU is beneficial to the overall C&TS design.

5.3 Establish Better Communication

The last and probably the most important recommendation is to improve the communication between all parties involved in software and hardware development for the C&TS. The poor state of the documentation led SwRI to believe that any current information related to the C&TS design must occur through informal discussions or messages. Presently, there does not appear to be sufficient technical interchange between contractors and NASA personnel to allow for efficient integration of all system

components. As a result, there is a large time lag between official design changes and implementation of those changes by the various contractors. The work done during the time lag is probably non-productive. There must be some organization actively expediting this interchange of information as part of the integration process. Without close communications between technical organizations responsible for the C&TS design the integration of the various elements and subsystems has little chance for success.

APPENDIX A

REVISED ASSESSMENT PLAN

REVISED ASSESSMENT PLAN
FOR

Research in Software Allocation
for Communications Tracking and Antennas
for Advanced Manned Missions
SwRI Project No. 05-3668

November 30, 1990

1.0 INTRODUCTION

This final revision of the preliminary assessment plan is intended as a general plan for the conduct of a software resources allocation study. The effort which produced this plan was a study of the software allocation problems associated with any communications and tracking system designed for manned spacecraft. The Communications and Tracking System of the Space Station Freedom was used as a test case and was the source for the methodology presented here.

2.0 PERTINENT DOCUMENTS COLLECTED

See Appendix B of the Final Report.

2.1 Project Plan Initial Task Descriptions

2.2 Task 1 - Initial Meeting/Present Data Collection Plan

Hold an initial meeting to present data collection plans, present this initial assessment plan, and discuss various program activities.

2.3 Task 2 - Develop Assessment Methodology

Develop a detailed assessment methodology by expanding this plan. The final plan should include feedback from NASA regarding the assessment plan, data collection arrangements, and details concerning the assessment methodology.

2.4 Task 3 - Collect Data

Collect all pertinent current information concerning the software and processing hardware under consideration and perform the following:

- Generate a summary of preliminary findings resulting from the data collection task

- Submit a draft plan for continuation of the resource assessment for NASA review and comment

- Perform the assessment upon NASA approval of the plan

2.5 Task 4 - Perform Assessment

Review and assess all collected data with the following primary objectives:

2.5.1

Identify all system processors (firmware and software controlled) and the following parameters:

- Architecture (bus structure, etc.)
- Memory size
- Execution Speed
- Resident CSCIs
- Throughput

2.5.2

Identify all system software and firmware and the following characteristics of each CSCI:

- Time & size requirements
- Function
- Host processor
- Programing language

2.5.3

Develop an understanding of all software CSCIs including :

- Software approach
- Interfaces to other CSCIs
- Alternate software approaches

2.5.4

Verify that planned software can be accommodated using planned hardware by:

- Matching contractor supplied software descriptions with requirements documents

- Checking the Time and Size requirements of each CSCI against actual available hardware and hardware architecture.

2.5.5

If the software and hardware are adequate, verify that the system requirements have not changed. If the software and hardware are inadequate, proceed as described in paragraph 2.4.7.

2.5.6

If the system requirements have not changed, conclude the assessment. If changes have occurred, proceed as described in paragraph 2.4.8.

2.5.7

If the conclusion reached in paragraph 2.4.5 is that the resources are not adequate, determine if they can be changed. If not, revise the requirements and proceed as described in paragraph 2.4.2. If the hardware and software can be changed, recommend the necessary changes and proceed as described in paragraph 2.4.6.

2.5.8

Determine the impact of any new requirements on existing hardware and software resources, then postulate new hardware and software designs. Proceed as described in paragraph 2.4.2.

2.6. Task 5 - Oral Presentation

After completion of Tasks 1 through 4 deliver an oral presentation summarizing the results of the assessment to cognizant NASA personnel. The presentation will be at a time and location proposed by the assessment organization and agreed to by NASA.

2.7 Task 6 - Prepare Final Report

Prepare a final report as described by the contract and the final assessment plan.

APPENDIX B

COMMUNICATIONS AND TRACKING DOCUMENTATION RECEIVED BY SwRI FROM NASA/JSC

APPENDIX B

COMMUNICATIONS AND TRACKING DOCUMENTATION RECEIVED BY SwRI FROM NASA/JSC

GENERAL DOCUMENTS

Document Title	Originator
Verification & Validation Plan (MDSSC-SSD-Houston), MDC H4249 April 1990, Resubmittal 1	MDSSC Space Station Div.
Contract NAS 9-18200, Software Test Plan (C&TS Firmware-Motorola), Work Package No. 2 (WP-2), (DR SY-36.1), May 1990	MDSSC Space Station Div.
Contract NAS 9-18200, Software Development Plan (GE), Appendix II, Work Package No. 2 (WP-2), (DR SY-26.1), May 1990	MDSSC Space Station Div.
Contract NAS 9-18200, Software Development Plan (GE), Appendix III, Work Package No. 2 (WP-2), (DR SY-26.1), May 1990	MDSSC Space Station Div.
Contract NAS 9-18200, Software Verification & Validation Plan (G.E.), Appendix I, Work Package No. 2 (WP-2), (DR SY-42.1), May 1990	MDSSC Space Station Div.
Space Station Software Standards Document, JSC 30244, version 3.0, September 19, 1986	Johnson Space Center
Contract End Item Specification for Communications and Tracking System (C&TS), (DR SY-06.1), May 1990	MDSSC Space Station Div.
Software Estimates and the Bilateral Agreement, July 3, 1990	NASA
Contract NAS 9-18200, Software Engineering & Integration Plan (C&TS-Control and Monitor Subsystem), (DR SY-29.1), November 1989	MDSSC Space Station Div.

GENERAL DOCUMENTS (Cont'd)

Document Title	Originator
Interface Requirements Document (Software) (DMS MIL-STD-1553B Local Bus), (DR SY-23.1), August 1990	MDSSC Space Station Div.
Contract NAS 9-18200, Software Management Plan, (DR SY-03.1), July 1990	MDSSC Space Station Div.
Software Process for Space Shuttle Primary Avionics Software System and Support Software, October 11, 1990	IBM
Space Station Communications & Tracking System (SS CTS), B1 Prime Item Development Specification Control and Monitor Subsystem, September 1990	GE Aerospace
User's Guide (C&T Control & Monitor/Orbital Replacement Unit Simulator), GE SY-40-002-STI, May 1990	MDSSC Space Station Div.
Software Preliminary Design Document (C&TS Control & Monitor Orbital Replaceable Unit Simulator, GE SY-28-003-STI, May 1990	MDSSC Space Station Div.
Systems Engineering and Integration Trade Studies - ORU Pressurization, SY-01.3-013, May 1990	MDSSC Space Station Div.
Processor Resource Allocation Document (DR SY-24.1), MDC H4372, September 1989	MDSSC Space Station Div.
Processor Resource Allocation Document (DR SY-24.1), MDC H4372, Resubmittal 2, May 1990	MDSSC Space Station Div.
Block Update to C&T Architectural Control Document, SSP 30260, November 21, 1989	MDSSC Space Station Div.

GENERAL DOCUMENTS (Cont'd)

Document Title	Originator
Contract NAS 9-18200, Interface Development Document (Communications & Tracking System Standard Interfaces) IDD 2CT00001, Work Package No. 2 (WP-2), (DR SY-49.1), April 1990	MDSSC Space Station Div.
Space Station Freedom Program Communications and Tracking System, Systems Engineering and Integration Trade Studies Control and Monitor Subsystem Architecture Options for Phase Assembly Sequence, SY-01.3-015, 31 May 1990	GE Aerospace
Electromagnetic, Plasma and Ionizing Radiation Effects Control Plan, (DR SY-17.1), Doc. #SY-17.1-002, September 1989	MDSSC Space Station Div.
Evaluation of the Lynxos Kernal Job Order 34-L10, LESC-28013, March 1990	NASA, Lockheed Engineering and Sciences Company
Justification for the Space Station BSPE Software Development Environment, PDRD-003S, January 1990	MDSSC Space Station Div.
Justification for the Space Station BSPE Software Development Environment, DPB-003 January 1990	MDSSC Space Station Div.
Critical Items List (CIL) for Communications and Tracking System, MDC H4918, May 1990	MDSSC Space Station Div.
Component AS-Designed EEE (Electrical, Electronic, and Electromechanical) Parts List, GE SA-10-001, April 12, 1990	MDSSC Space Station Div.
Processor Resource Allocation Document, MDC H4372, Resubmittal #1, December 1989	MDSSC Space Station Div.

GENERAL DOCUMENTS (Cont'd)

Document Title	Originator
Preliminary, Radio Frequency Interface Control Document Between the Space Station Freedom Project and the Tracking and Data Relay Satellite System, SSP 42018, April 30, 1990	MDSSC Space Station Div.
Communications and Tracking System Subsystem Design Review, November 13-17, 1989 (Briefing viewgraphs)	GE Aerospace
Contract End Item Specification for Communication and Tracking System (C&TS), SP-M-002, May 1990	MDSSC Space Station Div.
SRU Design Document for the Embedded Controller, 562-SSF-SGVBSP-061, January 1990	MDSSC Space Station Div.
Processor Resource Allocation Document dated May 1990, MDC H4372	MDSSC Space Station Div.
Space Station C&T System Preliminary Design Review Summary Presentation Day 1 only (5 day PDR), May 30, 1990	MDSSC Space Station Div.
Space Station C&T System Preliminary Design Review Summary Presentation Day 2 only (5 day PDR), May 30, 1990	MDSSC Space Station Div.
(Day 5 Only) Space Station C&T System Preliminary Design Review Summary Presentation May 30 to June 5, 1990	MDSSC Space Station Div.

SPACE-TO-SPACE SUBSYSTEM DOCUMENTS

Document Title	Originator
Software Requirements Specification (C&T SSS Modem), Spec No. 10033264, March 1990	MDSSC Space Station Div.
Software Requirements Specification (C&T - SSS Video Bandband Signal Processor), Spec. No. 10033494, March 1990	MDSSC Space Station Div.
Software Requirements Specification (C&T - SSS Transmitter - Receiver), Spec. No. 10033265, March 1990	MDSSC Space Station Div.
Software Requirements Specification (C&T SSS IF Switch), Spec No. 10033262, March 1990	MDSSC Space Station Div.

SPACE-TO-GROUND SUBSYSTEM DOCUMENTS

Document Title	Originator
Software Requirements Specification (C&T - SGS Video Baseband Signal Processor), Spec. No. 10033269, March 1990	MDSSC Space Station Div.
Contract NAS 9-18200, Software Requirements Specification (C&TS Firmware-SGS Antenna Controller), Work Package No. 2 (WP-2), (DR SY-34.1), May 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Digital Video Switch of the Space-to-Ground Video Baseband Signal Processor, 562-SSF-SGVBSP-051, February 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Video Input Formatter of the Space-to-Ground Video Baseband Signal Processor, 562-SSF-SGVBSP-041, February 1990	MDSSC Space Station Div.

SPACE-TO-GROUND SUBSYSTEM DOCUMENTS (Cont'd)

Document Title	Originator
Configuration Item Interconnection Definition Document for the Space-to-Ground Video Baseband Signal Processor, 562-SSF-SGVBSP-005, February 1989	MDSSC Space Station Div.
Software Preliminary Design Document for the Space-to-Ground Baseband Signal Processor, Preliminary, DPB-001, January 1990	MDSSC Space Station Div.
Ground Baseband Signal Processor (SGBSP), Preliminary, DPB-004, January 1990	MDSSC Space Station Div.
Software Requirements Specification (C&T SGS Transmitter - Receiver), Spec. No. 10033273, March 1990	MDSSC Space Station Div.
SGBSP Analysis, SSP-021, October 1989	MDSSC Space Station Div.
Software Requirements Specification (C&T - SGS IF Switch), Spec. No. 10033272, March 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Bite & Sync Detector of the Space-to-Ground Video Baseband Signal Processor, 562-SSF-SGVBSP-071, February 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Video A/D Converter of the Space-to-Ground Video Signal Processor, 562-SSF-SGVBSP-011, February 1989	MDSSC Space Station Div.
Lower Level Configuration Item Specification - Space-to-Ground Subsystem, Prelim-2, 10032303, April 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Video Output Formatter of the Space-to-Ground Video Baseband Signal Processor, 562-SSF-SGVBSP-031	MDSSC Space Station Div.

SPACE-TO-GROUND SUBSYSTEM DOCUMENTS (Cont'd)

Document Title	Originator
Lower Level Configuration Item Specification - Video Baseband Signal Processor Type 1, 2, 10033040, April 1990	MDSSC Space Station Div.
SGS Antenna Group Type I & II, RML-009-89-101, December 1989	MDSSC Space Station Div.
High Rate Modem (HRM), Preliminary, DPB-008, January 1990	MDSSC Space Station Div.
Critical Item Allocation & Partitioning Document for the Space-to-Ground Baseband Signal Processor, 562-SSF-SGVBSP-002, February 1989	MDSSC Space Station Div.
Lower Level Configuration Item Specification-Space-to-Ground Subsystem, Prelim-2, (DR SY-06.2), 10032303, April 1990	MDSSC Space Station Div.

VIDEO SUBSYSTEM DOCUMENTS

Document Title	Originator
Software Requirements Specification (C&T Video Switch, Spec. No. 10033706, March 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Crosspoint Switch Matrix (CSM) of the Video Switch, 562-SSF-VSW-021, February 1990	MDSSC Space Station Div.
Video Subsystem Support Documentation-Integration and Standardization Bulletin #1, SS/C&T-561-IC-1742B, October 1989	MDSSC Space Station Div.
Lower Level Configuration Item Specification- Pan Tilt Unit (PTU), 10033023, April 1990	MDSSC Space Station Div.

VIDEO SUBSYSTEM DOCUMENTS (Cont'd)

Document Title	Originator
Allocation & Partitioning Document for the Television Camera Interface Converter of the External Color TV Camera Group, 562-SSF-TVCIC-002, February 1989	MDSSC Space Station Div.
Lower Level Configuration Item Specification-Source Control Drawing Color TV Camera, 10033497, April 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the PFM Modulator/PFM Demodulator Fiber Optic Transmitter and Fiber Optic Receiver of the Video Switch, 562-SSF-VSW-051, February 1990	MDSSC Space Station Div.
Critical Item Interconnection Definition Document for the Video Switch, 562-SSF-VSW-005, February 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Data Stripper Board (DSB) of the Video Switch, 562-SSF-VSW-041, February 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Sync/Control/Bite (SCB) Board of the Video Switch, 562-SSF-VSW-011, February 1990	MDSSC Space Station Div.
Lower Level Configuration Item Specification -TV Camera Interface Converter, 10033125, April 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Basic Signal Generator (BSG) of the Video Switch, 562-SSF-VSW-031, February 1990	MDSSC Space Station Div.
Critical Item Allocations and Partitioning Document Video Subsystem Video Switch, 562-SSF-VSW-002, February 1990	MDSCC Space Station Div.

VIDEO SUBSYSTEM DOCUMENTS (Cont'd)

Document Title	Originator
Lower Level Configuration Item Specification Video Switch, 10033705, May 1990	MDSSC Space Station Div.
SRU Electrical Design Document for the Built-In Test Board of the Television Camera Interface Converter, 562-SSF-TVCIC-011, February 1989	MDSSC Space Station Div.
Lower Level Configuration Item Specification -Video Subsystem Fiber Optic Transmitter and Fiber Optic Receiver, 10033031, May 1990	MDSSC Space Station Div.
Lower Level Configuration Item Specification -External TV Camera Group, 10033124, April 1990	MDSSC Space Station Div.
Lower Level Configuration Item Specification -Video Subsystem PFM Modulator and PFM Demodulation, Prelim-O, 10033047, May 1990	MDSSC Space Station Div.

CONTROL & MONITOR SUBSYSTEM DOCUMENTS

Document Title	Originator
Configuration Item Development Specification for the Embedded Data Processor, (DR SY-06.2), MDC H4534 November 1989	MDSSC Space Station Div.
Configuration Item Development Specification for the Bus Network Interface Unit, Work Package No. 2 (WP-2), (DR SY-06.2), November 1989	MDSSC Space Station Div.
Configuration Item (CI) Specification for Standard Data Processor (DR SY-06.2), 152A401-PT1A, May 1990	NASA

CONTROL & MONITOR SUBSYSTEM DOCUMENTS

Document Title	Originator
Interface Requirements Document (Software) (C&TS - Control & Monitor Processor Application), Resubmittal, MDC H4418, May 1990	MDSSC Space Station Div.
Interface Requirements Document (Software) Appendix I - Interface Control Document (C&TS - Local Controller Application) DR SY-23.1, MDC H4417, May 1990	MDSSC Space Station Div.
Interface Requirements Document (Software) (C&TS - Local Controller Application) DR SY-23.1, MDC H4417, May 1990	MDSSC Space Station Div.
Interface Requirements Document (Software), Appendix I - Interface Control Document, (C&TS - Local Controller Application), DR SY-23.1, MDC H4417, Appendix I, May 1990	MDSSC Space Station Div.
Software Requirements Specification (C&T - Control & Monitor Processor Application), MDC H4416, May 1990	MDSSC Space Station Div.
Software Requirements Specification (C&T - Control & Monitor/Orbital Replacement Unit Simulator), MDC H4524, May 1990	MDSSC Space Station Div.
Software Requirements Specification (C&T - Local Controller Application), MDC H4415, May 1990	MDSSC Space Station Div.
Software Preliminary Design Document (C&TS Control & Monitor Processor Application), GE SY-33-001A, May 1990	MDSSC Space Station Div.
Software Preliminary Design Document (C&TS Local Controller Application), SY-33-002, May 1990	MDSSC Space Station Div.

DATA MANAGEMENT SYSTEM DOCUMENTS

Document Title	Originator
Runtime Object Database (RODB) Sizing White Paper, July 20, 1990, David E. Bolon	International Business Machines Corp.
Software Requirements Specification (DMS Standard Services) (DR SY-34.1), August 1990	MDSSC Space Station Div.
Software Detailed Design Document (Data Management System Standard Services), (DR SY-28.1), September 1990	NASA
Software Preliminary Design Document (DMS Operating System/ADA Run Time Environment, DR SY-33.1), June 1990	NASA
Space Station Program Data Management System,Systems Engineering And Integration Trade Studies DMS Performance Analysis Summary White Paper, Contract No. 87916006, DR SY-01.31, July 31, 1990	IBM
DMS Performance, September 26, 1990	IBM
Architectural Control Document Data Management System Section 2: Operations Management System, Revision C, SSP 30261 Sec. 2, December 15, 1989	NASA
Configuration Item (CI) Specification for Mass Storage Unit, MDC H4686, (DR SY-06.2), March 1990	MDSSC Space Station Div.
Contract NAS 9-18200, Data Management System to Communications and Tracking System Interface Development Document, Work Package No. 2, (WP-2), March 1990	MDSSC Space Station Div.
Configuration Item (CI) Specification for Mass Storage Unit, (DR SY-06.2), Spec. #153A101, FSCM #18355, March 1990	MDSSC Space Station Div.

DATA MANAGEMENT SYSTEM DOCUMENTS (Cont'd)

Document Title	Originator
Forward Error Correction for Transferred Data Frames, SSP-95, November 1989	MDSSC Space Station Div.
Data Management System (DMS) S/W Orientation Handbook	MDSSC Space Station Div.
Data Management System (DMS) H/W Orientation Handbook	MDSSC Space Station Div.
EXCERPT from Summary Presentation DMS PDR No. 3 Day 1 April 17, 1990, MDC H4866	MDSSC Space Station Div.
Configuration Item Development Specification for the Bus Network Interface Unit, DR SY-06.2, MDC H4533, November 1989	MDSSC Space Station Div.
Users Guide (Software), Data Management System (IBM), DR SY-40.1, MDC H4542, December 1989	MDSSC Space Station Div.
Critical Item Development Specification for the Standard Data Processor, 88IBMX0069-RC, June 16, 1989	MDSSC Space Station Div.

UHF SUBSYSTEM DOCUMENTS

Document Title	Originator
Lower Level Configuration Item Specification - UHF Communications Subsystem, Draft-1, 10033487, April 1990	MDSSC Space Station Div.

ASSEMBLY/CONTINGENCY SUBSYSTEM DOCUMENTS

Document Title	Originator
Lower Level Configuration Item Specification-Assembly/Contingency Baseband Signal Processor (ACBSP), 10032331, January 1990	MDSSC Space Station Div.
Section 4.0 of the Software Requirements Specification for the Standard TDRSS Transponder, Preliminary, DPB-007, January 1990	MDSSC Space Station Div.

TRACKING SUBSYSTEM DOCUMENTS

Document Title	Originator
Space Station Freedom Program Radio Frequency Interface Control Document, Space Station Freedom Project to Global Positioning System (SSP 42017), June 1990	MDSSC Space Station Div.